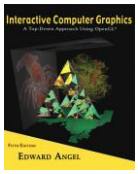


Name: _____ Term # ____

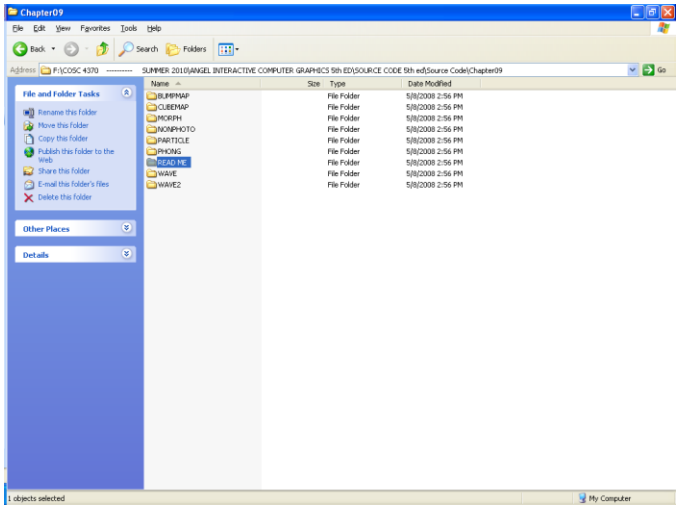
Homework 9 SOLUTIONS

(400 points)

NOTE: **Chapter 9 of the textbook shows the usage of the vertex and fragment programmable shaders.**
Part B is an OpenGL application.

Visual Studio 2008 C++ **Project**

Create Visual Studio 2008 C++, Empty Project, Homework9:
For each of the following **8 programmable shaders** and their variations



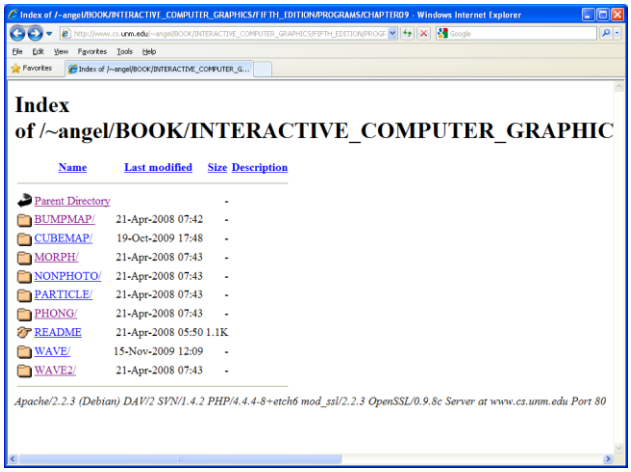
Build and run Project &

1. Besides the identification of the GLSL example & the screenshot of it's output:

2. Print it's v and f shaders and explain what it does.

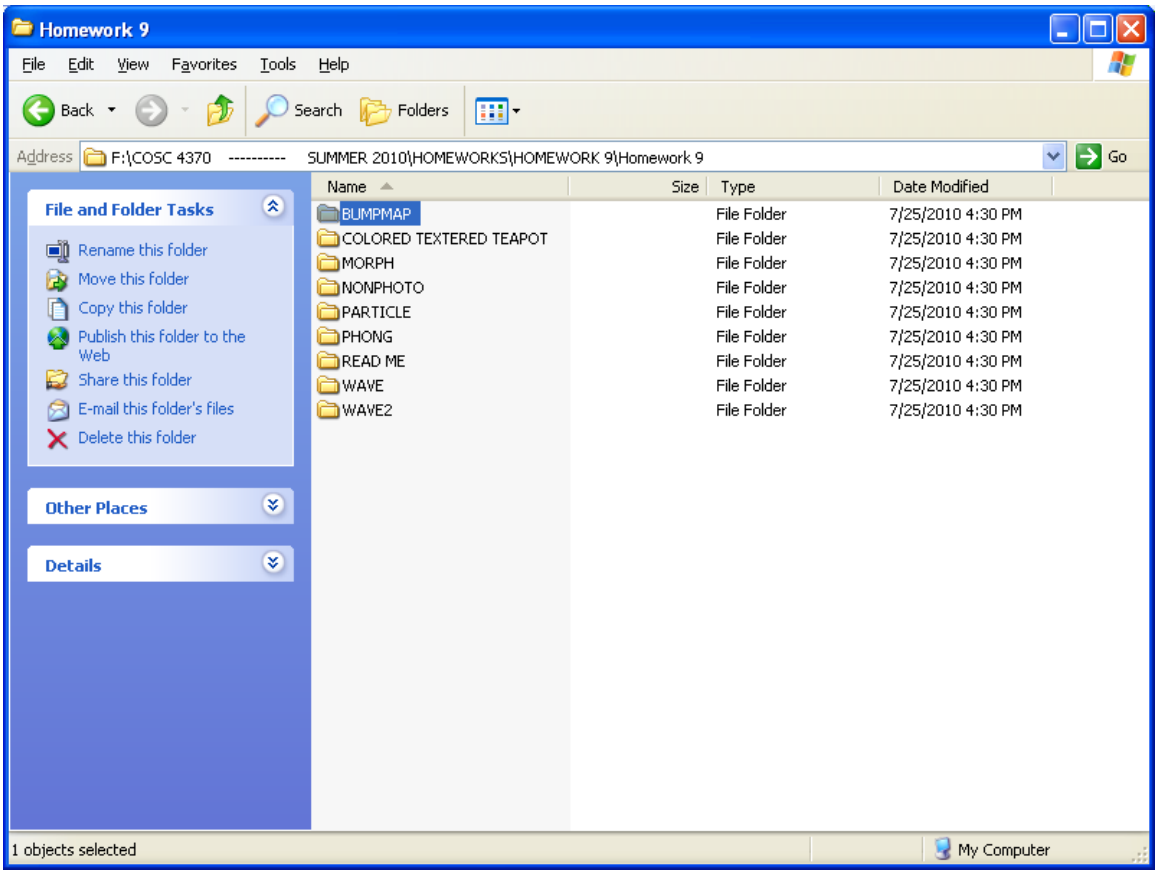
3. You need to include the parts of the .c program without the shader's reading or initialization parts since this code is the same for ALL OpenGL code that uses shader programs.

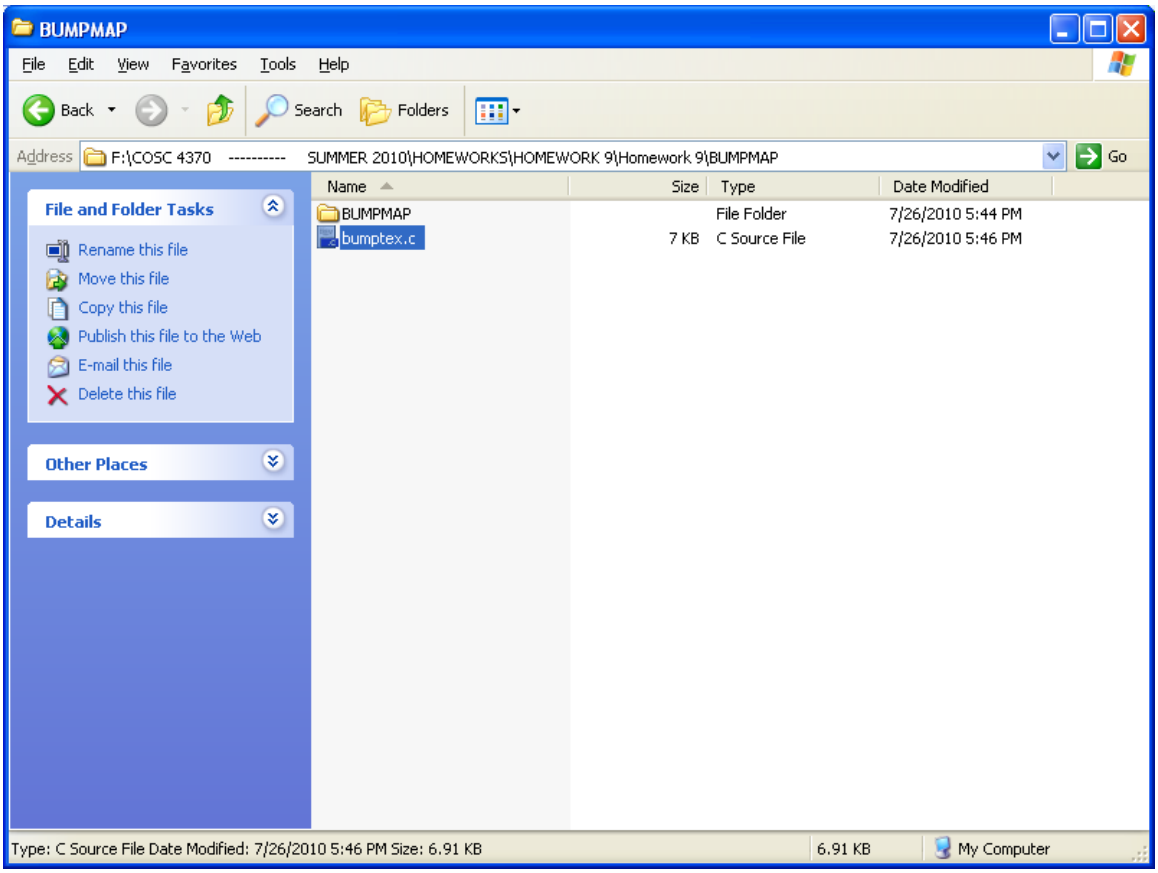
http://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/FIFTH_EDITION/PROGRAMS/CHAPTER09/

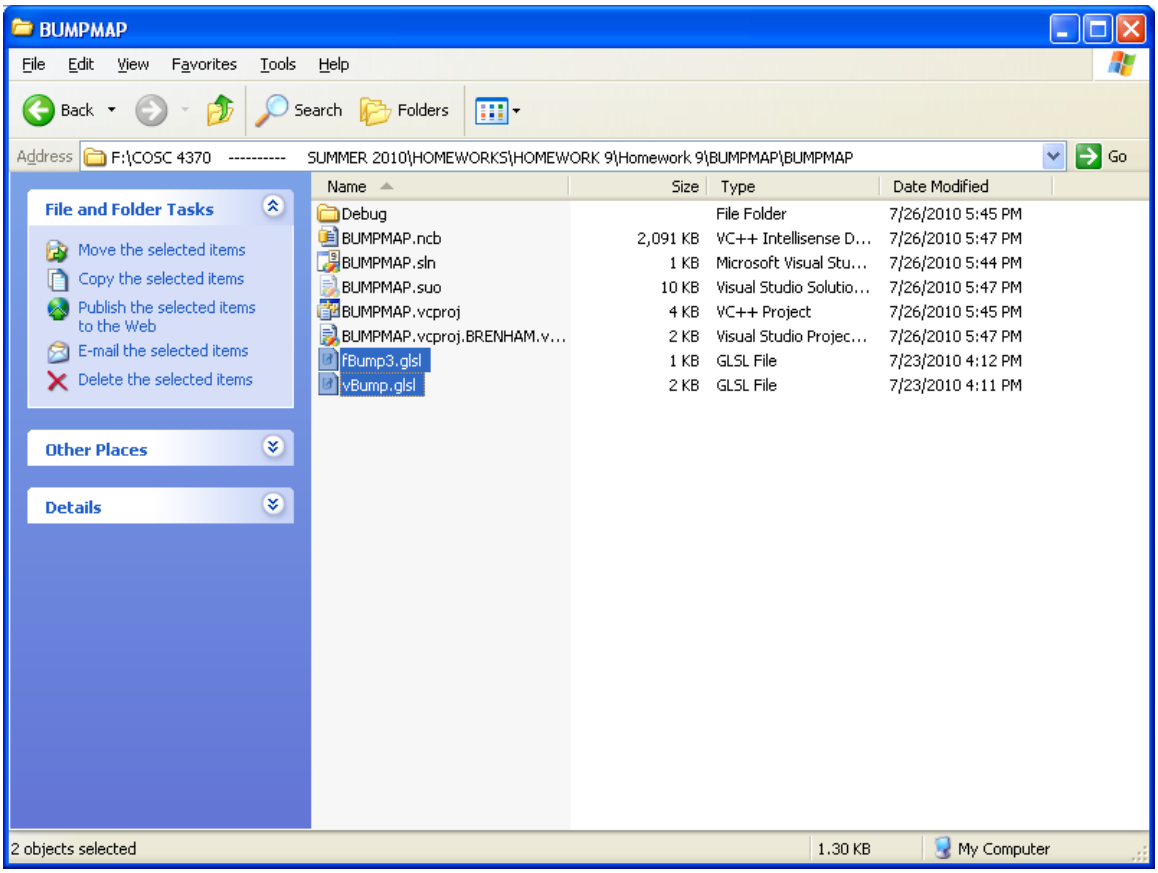


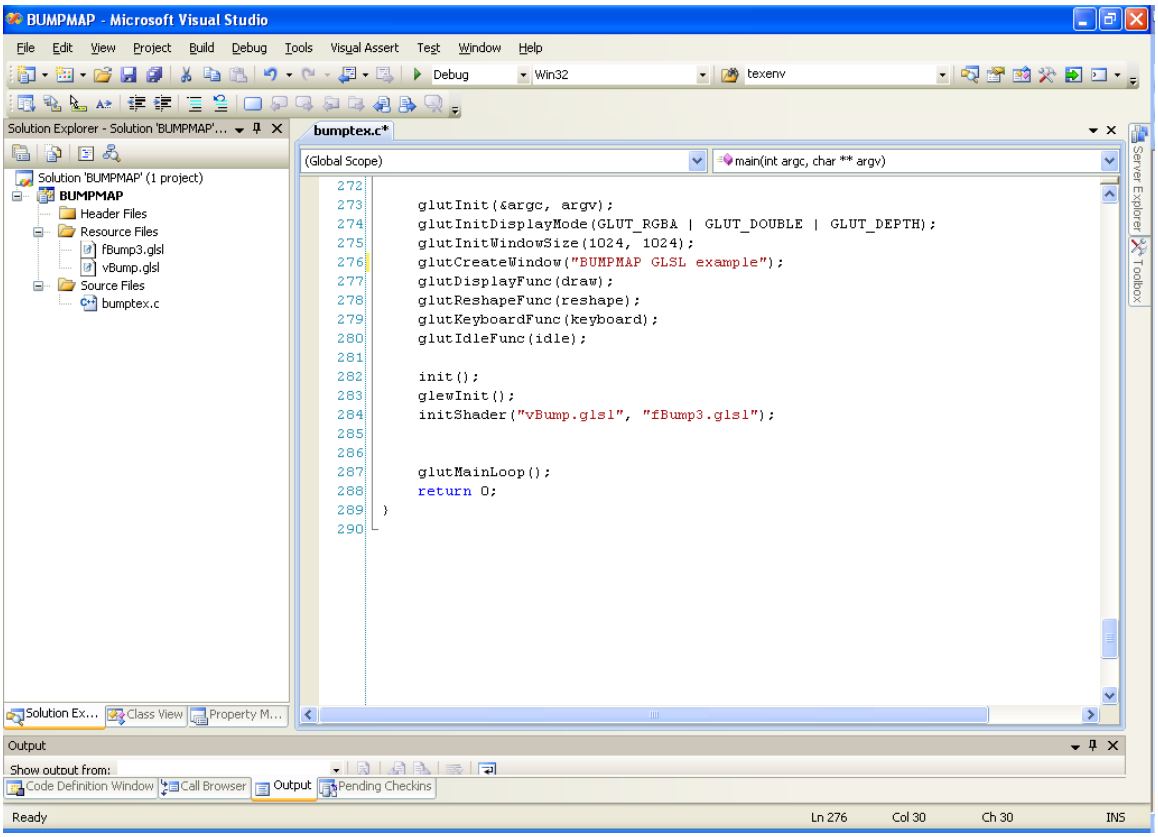
ANSWER:

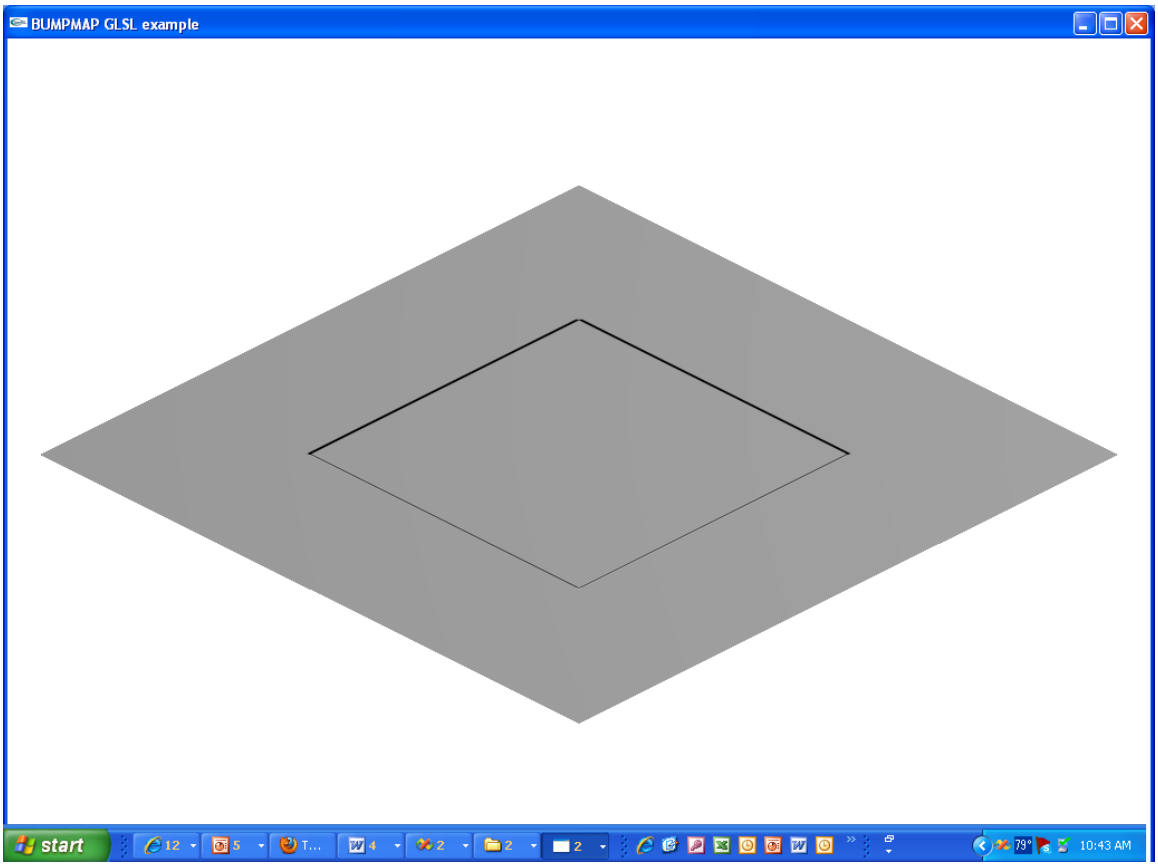
1. BUMPMAP


Homework 9
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9
Go
File and Folder Tasks
Rename this folder
Move this folder
Copy this folder
Publish this folder to the Web
Share this folder
E-mail this folder's files
Delete this folder
Other Places
Details
Name Size Type Date Modified
BUMPMAP File Folder 7/25/2010 4:30 PM
COLORED TEXTERED TEAPOT File Folder 7/25/2010 4:30 PM
MORPH File Folder 7/25/2010 4:30 PM
NONPHOTO File Folder 7/25/2010 4:30 PM
PARTICLE File Folder 7/25/2010 4:30 PM
PHONG File Folder 7/25/2010 4:30 PM
READ ME File Folder 7/25/2010 4:30 PM
WAVE File Folder 7/25/2010 4:30 PM
WAVE2 File Folder 7/25/2010 4:30 PM
1 objects selected
My Computer


BUMPMAP
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\BUMPMAP
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places
Details
Name Size Type Date Modified
BUMPMAP File Folder 7/26/2010 5:44 PM
bumptex.c 7 KB C Source File 7/26/2010 5:46 PM
Type: C Source File Date Modified: 7/26/2010 5:46 PM Size: 6.91 KB
6.91 KB
My Computer


BUMPMAP
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\BUMPMAP\BUMPMAP
Go
File and Folder Tasks
Move the selected items
Copy the selected items
Publish the selected items to the Web
E-mail the selected items
Delete the selected items
Other Places
Details
Name Size Type Date Modified
Debug File Folder 7/26/2010 5:45 PM
BUMPMAP.ncb 2,091 KB VC++ Intellisense D... 7/26/2010 5:47 PM
BUMPMAP.sln 1 KB Microsoft Visual Stu... 7/26/2010 5:44 PM
BUMPMAP.suo 10 KB Visual Studio Solutio... 7/26/2010 5:47 PM
BUMPMAP.vcproj 4 KB VC++ Project 7/26/2010 5:45 PM
BUMPMAP.vcproj.BRENHAM.v... 2 KB Visual Studio Projec... 7/26/2010 5:47 PM
fBump3.glsl 1 KB GLSL File 7/23/2010 4:12 PM
vBump.glsl 2 KB GLSL File 7/23/2010 4:11 PM
2 objects selected
1.30 KB
My Computer


BUMPMAP - Microsoft Visual Studio
File Edit View Project Build Debug Tools Visual Assert Test Window Help
Debug Win32 texenv
Solution Explorer - Solution 'BUMPMAP'...
Solution 'BUMPMAP' (1 project)
BUMPMAP
Header Files
Resource Files
fBump3.glsl
vBump.glsl
Source Files
bumptex.c
bumptex.c*
(Global Scope)
main(int argc, char ** argv)
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
glutInitWindowSize(1024, 1024);
glutCreateWindow("BUMPMAP GLSL example");
glutDisplayFunc(draw);
glutReshapeFunc(reshape);
glutKeyboardFunc(keyboard);
glutIdleFunc(idle);
init();
glewInit();
initShader("vBump.glsl", "fBump3.glsl");
glutMainLoop();
return 0;
}
Output
Show output from:
Code Definition Window
Call Browser
Output
Pending Checkins
Ready
Ln 276
Col 30
Ch 30
INS




```
// bumptex.c

/* sets up flat mesh */
/* sets up elapsed time parameter for use by shaders */



#include <stdio.h>
#include <GL/glew.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <GL/glut.h>
#include <math.h>

#define N 256
#define a 0.1
#define b 50.0

GLfloat normals[N][N][3];
const GLdouble nearVal     = 1.0;
const GLdouble farVal      = 20.0;
GLfloat  lightPos[4] = {0.0, 10.0, 0.0, 1.0};
GLuint         program     = 0;
/* GLint          timeParam; */
GLuint        texHandle;
GLuint        texMapLocation;
GLint          tangentParam;
```

```
const GLfloat tangent[3] = {1.0, 0.0, 0.0};

/* shader reader */
/* creates null terminated string from file */

static char* readShaderSource(const char* shaderFile)
{
    FILE* fp = fopen(shaderFile, "rb");

    char* buf;
    long size;

    if(fp==NULL) return NULL;
    fseek(fp, 0L, SEEK_END);
    size = ftell(fp);
    fseek(fp, 0L, SEEK_SET);
    buf = (char*) malloc((size+1) * sizeof(char));
    fread(buf, 1, size, fp);
      buf[size] = '\0';
    fclose(fp);
    return buf;
}

/* error printing function */

static void checkError(GLint status, const char *msg)
{
    if (status == GL_FALSE)
    {
        printf("%s\n", msg);
        exit(EXIT_FAILURE);
    }
}

/* standard OpenGL initialization */

static void init()
{
    const float meshColor[]     = {0.7f, 0.7f, 0.7f, 1.0f};
    const float meshSpecular[]  = {0.8f, 0.8f, 0.8f, 1.0f};
    const float meshShininess[] = {80.0f};

    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, meshColor);
    glMaterialfv(GL_FRONT, GL_SPECULAR, meshSpecular);
    glMaterialfv(GL_FRONT, GL_SHININESS, meshShininess);

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

    glGenTextures(1, &texHandle);
    glBindTexture(GL_TEXTURE_2D, texHandle);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, N, N, 0, GL_RGB, GL_FLOAT,
normals);
     glEnable(GL_TEXTURE_2D);
```

```
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-0.75,0.75,-0.75,0.75,-5.5,5.5);

    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
     glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
}

/* GLSL initialization */

static void initShader(const GLchar* vShaderFile, const GLchar*
fShaderFile)
{
    GLint status;
    GLchar *vSource, *fSource;
    GLuint vShader, fShader;

    /* read shader files */

    vSource = readShaderSource(vShaderFile);
    if(vSource==NULL)
    {
        printf( "Failed to read vertex shaderi\n");
        exit(EXIT_FAILURE);
    }

    fSource = readShaderSource(fShaderFile);
    if(fSource==NULL)
    {
        printf("Failed to read fragment shader");
        exit(EXIT_FAILURE);
    }

    /* create program and shader objects */

    vShader = glCreateShader(GL_VERTEX_SHADER);
    fShader = glCreateShader(GL_FRAGMENT_SHADER);
    program = glCreateProgram();

    /* attach shaders to the program object */

    glAttachShader(program, vShader);
    glAttachShader(program, fShader);

    /* read shaders */

    glShaderSource(vShader, 1, (const GLchar**) &vSource, NULL);
    glShaderSource(fShader, 1, (const GLchar**) &fSource, NULL);

    /* compile shaders */

    glCompileShader(vShader);
    glCompileShader(fShader);

    /* error check */
```

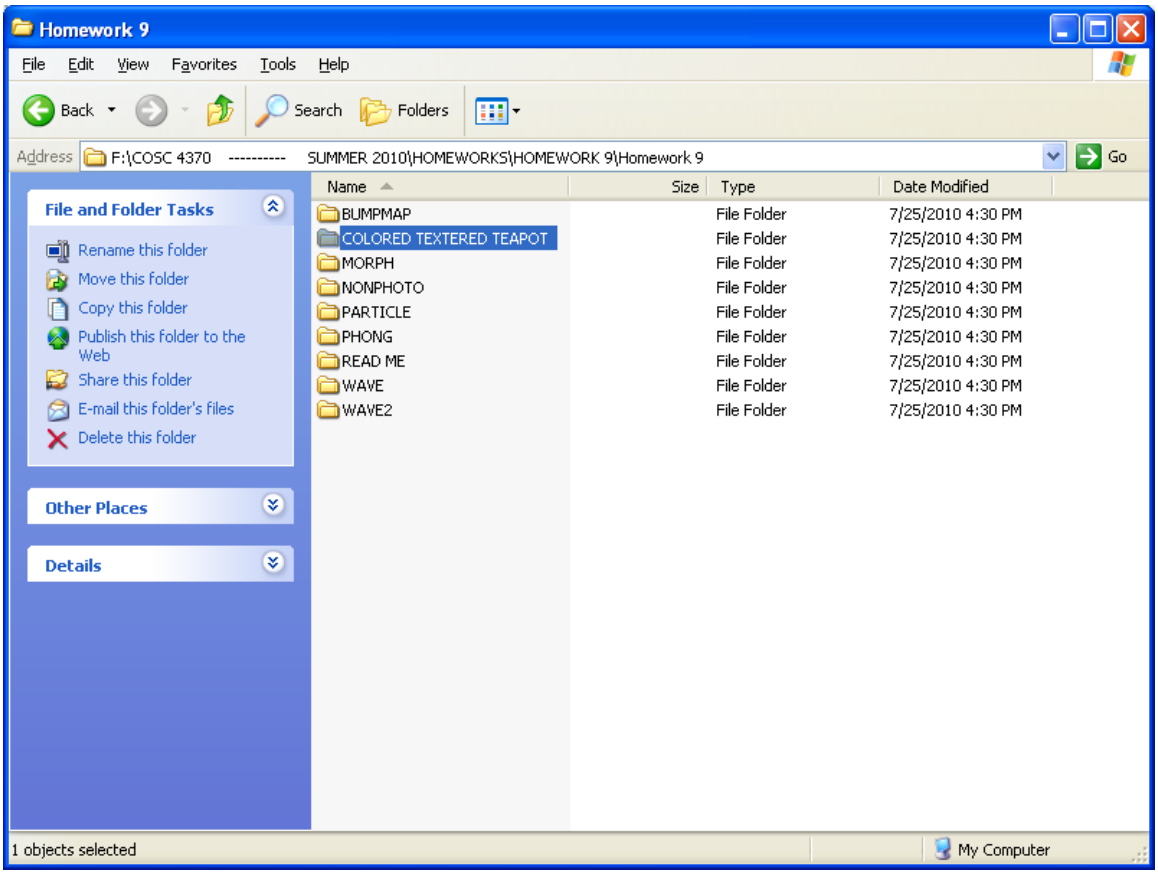
```
    glGetShaderiv(vShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the vertex shader.");

    glGetShaderiv(fShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the fragment shader.");

    /* link */

    glLinkProgram(program);
    glGetProgramiv(program, GL_LINK_STATUS, &status);
    checkError(status, "Failed to link the shader program object.");

    /* use program object */

    glUseProgram(program);

    /* set up uniform parameter */

    tangentParam = glGetAttribLocation(program, "objTangent");
    /* timeParam = glGetUniformLocation(program, "time"); */
    texMapLocation = glGetUniformLocation(program, "texMap");
}

void mesh()
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(2.0, 2.0, 2.0, 0.5, 0.0, 0.5, 0.0, 1.0, 0.0);
       glNormal3f(0.0,1.0,0.0);
       glBegin(GL_QUADS);
       glTexCoord2f(0.0, 0.0);
       glVertex3f(0.0, 0.0, 0.0);
       glTexCoord2f(1.0, 0.0);
       glVertex3f(1.0, 0.0, 0.0);
       glTexCoord2f(1.0, 1.0);
       glVertex3f(1.0,0.0,1.0);
       glTexCoord2f(0.0, 1.0);
       glVertex3f(0.0,0.0,1.0);
       glEnd();
}

static void draw(void)
{

    /* glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME)); */

    glVertexAttrib3fv(tangentParam, tangent);
    glUniform1i(texMapLocation, 0);

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
     mesh();
    /* glutSolidTeapot(0.4); */
    glutSwapBuffers();
}

static void reshape(int w, int h)
```

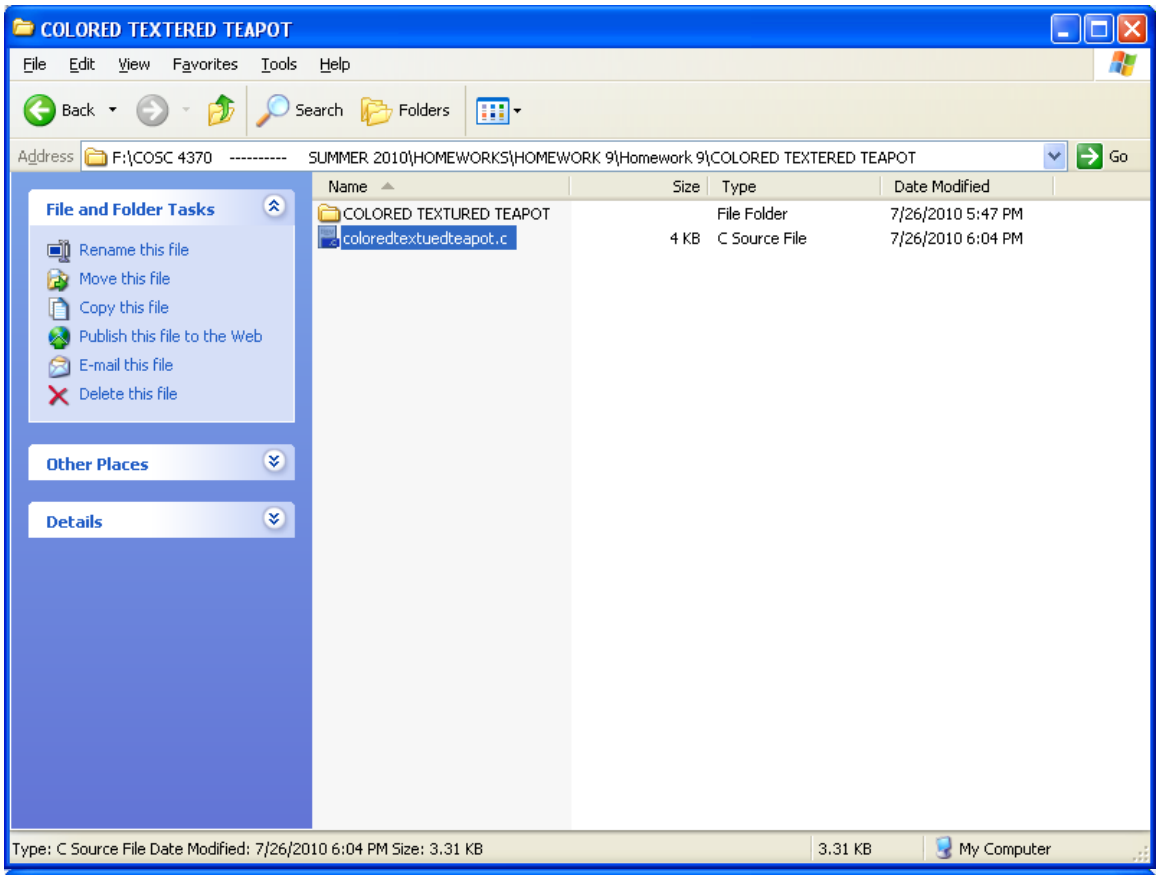
```
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-0.75,0.75,-0.75,0.75,-5.5,5.5);

    glViewport(0, 0, w, h);
    glutPostRedisplay();
}

static void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 27:
    case 'Q':
    case 'q':
        exit(EXIT_SUCCESS);
        break;
    case '1':
         lightPos[0] = 10.0;
         lightPos[1] = 10.0;
         lightPos[2] = 0.0;
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
         glutPostRedisplay();
         break;
    case '2':
         lightPos[0] = 7.0;
         lightPos[1] = 10.0;
         lightPos[2] = 7.0;
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
         glutPostRedisplay();
         break;
    default:
        break;
    }
}

void idle()
{
   int t;
   /* glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME)); */
   t = glutGet(GLUT_ELAPSED_TIME);
   lightPos[0] = 5.5*sin(0.001*t);
   lightPos[2] = 5.5*cos(0.001*t);
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
   glutPostRedisplay();
}

int main(int argc, char** argv)
{
    int i,j,k;
    float d;

    float data[N+1][N+1];
    for(i=0;i<N+1;i++) for(j=0;j<N+1;j++) data[i][j]=0.0;
    for(i=N/4; i< 3*N/4; i++) for(j=N/4;j<3*N/4;j++) data[i][j] =
100.5;
```

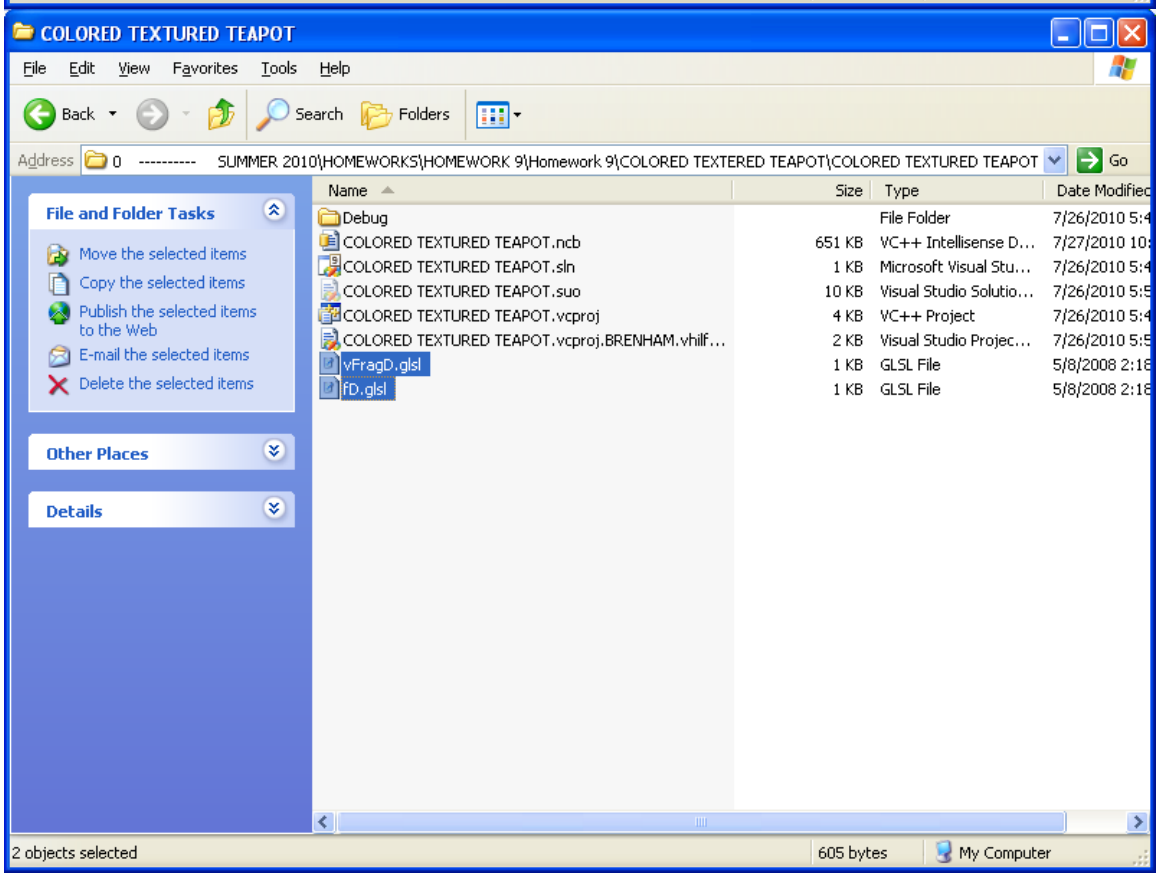
```
    for(i=0;i<N;i++) for(j=0;j<N;j++)
    {
       normals[i][j][0] =
a*b+sin(b*i/(float)N)*(float)cos(b*j/(float)N);
       normals[i][j][1] = a*b+sin(b*j/(float)N)*cos(b*i/(float)N);
       normals[i][j][2]=1.0;
    }
    for(i=0;i<N;i++) for(j=0;j<N;j++)
    {
       normals[i][j][0] = data[i][j]-data[i+1][j];
       normals[i][j][1] = data[i][j]-data[i][j+1];
       normals[i][j][2]=1.0;
    }
    for(i=0;i<N;i++) for(j=0;j<N;j++)
    {
       d = 0.0;
       for(k=0;k<3;k++) d+=normals[i][j][k]*normals[i][j][k];
       d=sqrt(d);
       for(k=0;k<3;k++) normals[i][j][k]= 0.5*normals[i][j][k]/d+0.5;
    }

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(1024, 1024);
    glutCreateWindow("BUMPMAP GLSL example");
    glutDisplayFunc(draw);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutIdleFunc(idle);

    init();
      glewInit();
    initShader("vBump.glsl", "fBump3.glsl");


    glutMainLoop();
    return 0;
}


// vBump.glsl
/* bump map vertex shader */

varying vec3 L; /* light vector in texture-space coordinates */
varying vec3 V; /* view vector in texture-space coordinates */
attribute vec3 objTangent; /* tangent vector in object coordinates */


void main()
{
    gl_Position = gl_ModelViewProjectionMatrix*gl_Vertex;


    gl_TexCoord[0] = gl_MultiTexCoord0;
    vec3 eyePosition = vec3(gl_ModelViewMatrix*gl_Vertex);
    vec3 eyeLightPos = vec3(gl_LightSource[0].position);
```

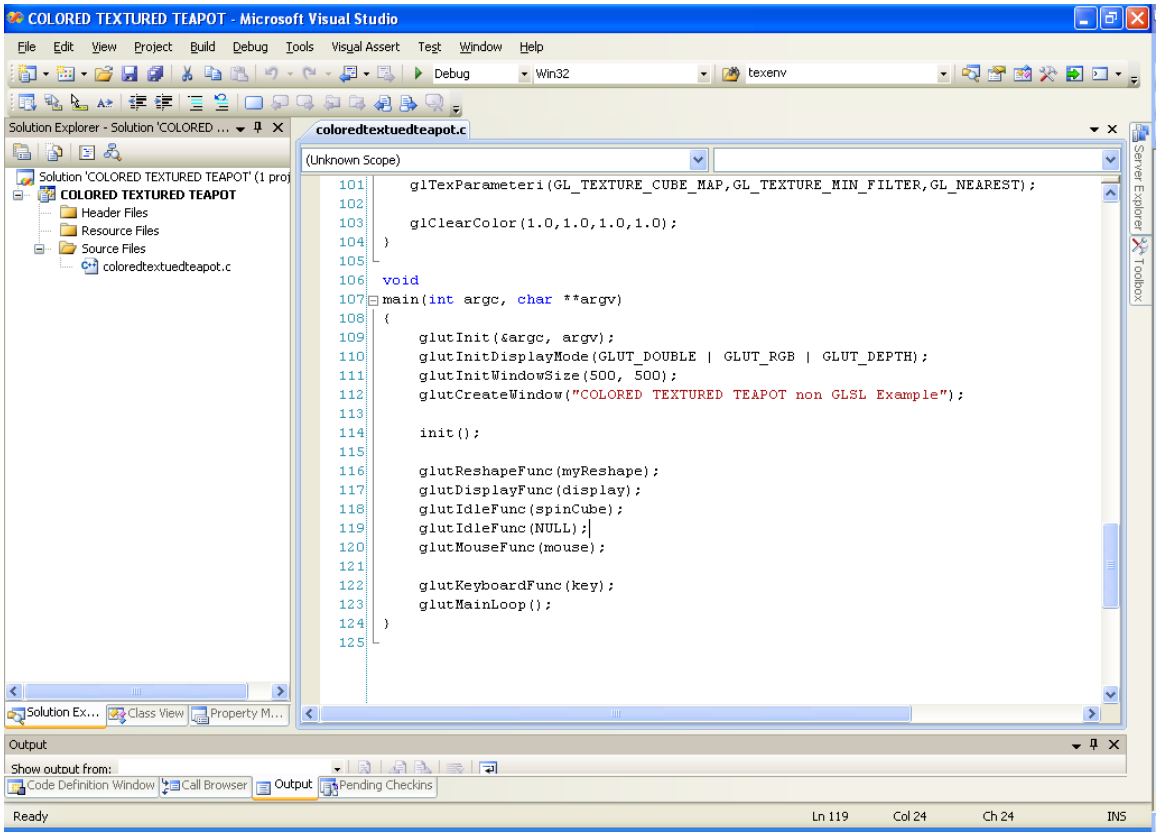
```
   /* normal, tangent and binormal in eye coordinates */

    vec3 N = normalize(gl_NormalMatrix*gl_Normal);
    vec3 T  = normalize(gl_NormalMatrix*objTangent);
    vec3 B = cross(N, T);

    /* light vector in texture space */

    L.x = dot(T, eyeLightPos-eyePosition);
    L.y = dot(B, eyeLightPos-eyePosition);
    L.z = dot(N, eyeLightPos-eyePosition);

    L = normalize(L);

    /* view vector in texture space */

    V.x = dot(T, -eyePosition.xyz);
    V.y = dot(B, -eyePosition.xyz);
    V.z = dot(N, -eyePosition.xyz);

    V = normalize(V);
}


// fBump3.glsl

varying vec3 L;
varying vec3 V;
uniform sampler2D texMap;

void main()
{

   vec4 N = texture2D(texMap, gl_TexCoord[0].st);
   vec3 NN =  normalize(2.0*N.xyz-1.0);
   vec3 LL = normalize(L);
  float Kd = max(dot(NN, LL), 0.0);
   gl_FragColor = Kd*gl_FrontLightProduct[0].diffuse;
}
```

2. COLORED TEXTURED TEAPOT




COLORED TEXTERED TEAPOT
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\COLORED TEXTERED TEAPOT
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places
Details
Name Size Type Date Modified
COLORED TEXTURED TEAPOT File Folder 7/26/2010 5:47 PM
coloredtextuedteapot.c 4 KB C Source File 7/26/2010 6:04 PM
Type: C Source File Date Modified: 7/26/2010 6:04 PM Size: 3.31 KB
3.31 KB
My Computer


COLORED TEXTURED TEAPOT
File Edit View Favorites Tools Help
Back Search Folders
Address 0 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\COLORED TEXTERED TEAPOT\COLORED TEXTURED TEAPOT
Go
File and Folder Tasks
Move the selected items
Copy the selected items
Publish the selected items to the Web
E-mail the selected items
Delete the selected items
Other Places
Details
Name Size Type Date Modified
Debug File Folder 7/26/2010 5:4
COLORED TEXTURED TEAPOT.ncb 651 KB VC++ Intellisense D... 7/27/2010 10:
COLORED TEXTURED TEAPOT.sln 1 KB Microsoft Visual Stu... 7/26/2010 5:4
COLORED TEXTURED TEAPOT.suo 10 KB Visual Studio Solutio... 7/26/2010 5:5
COLORED TEXTURED TEAPOT.vcproj 4 KB VC++ Project 7/26/2010 5:4
COLORED TEXTURED TEAPOT.vcproj.BRENHAM.vhilf... 2 KB Visual Studio Projec... 7/26/2010 5:5
vFragD.glsl 1 KB GLSL File 5/8/2008 2:18
fD.glsl 1 KB GLSL File 5/8/2008 2:18
2 objects selected
605 bytes
My Computer

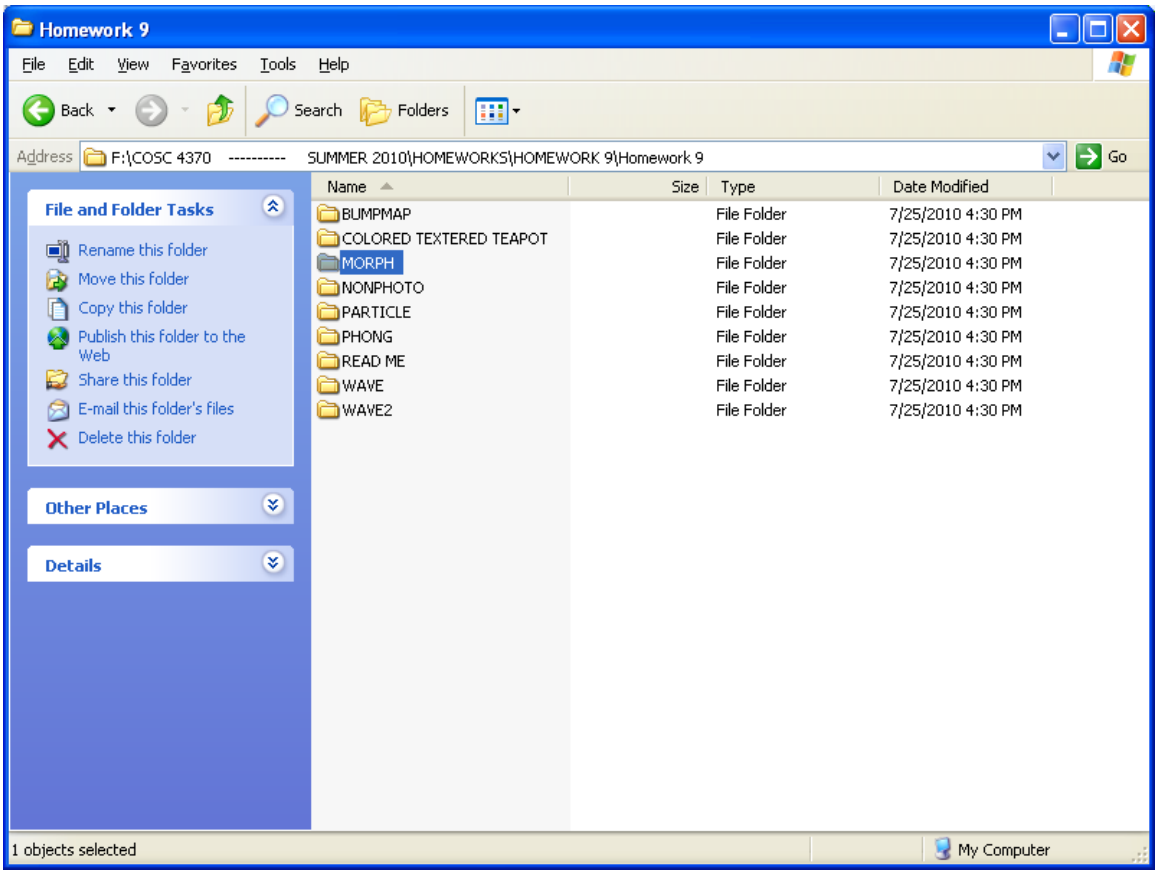

COLORED TEXTURED TEAPOT - Microsoft Visual Studio
File Edit View Project Build Debug Tools Visual Assert Test Window Help
Debug
Win32
texenv
Solution Explorer - Solution 'COLORED ...
Solution 'COLORED TEXTURED TEAPOT' (1 proj
COLORED TEXTURED TEAPOT
Header Files
Resource Files
Source Files
coloredtextuedteapot.c
coloredtextuedteapot.c
(Unknown Scope)
101 glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
102
103 glClearColor(1.0,1.0,1.0,1.0);
104 }
105
106 void
107 main(int argc, char **argv)
108 {
109 glutInit(&argc, argv);
110 glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
111 glutInitWindowSize(500, 500);
112 glutCreateWindow("COLORED TEXTURED TEAPOT non GLSL Example");
113
114 init();
115
116 glutReshapeFunc(myReshape);
117 glutDisplayFunc(display);
118 glutIdleFunc(spinCube);
119 glutIdleFunc(NULL);
120 glutMouseFunc(mouse);
121
122 glutKeyboardFunc(key);
123 glutMainLoop();
124 }
125
Solution Ex...
Class View
Property M...
Server Explorer
Toolbox
Output
Show output from:
Code Definition Window
Call Browser
Output
Pending Checkins
Ready
Ln 119
Col 24
Ch 24
INS

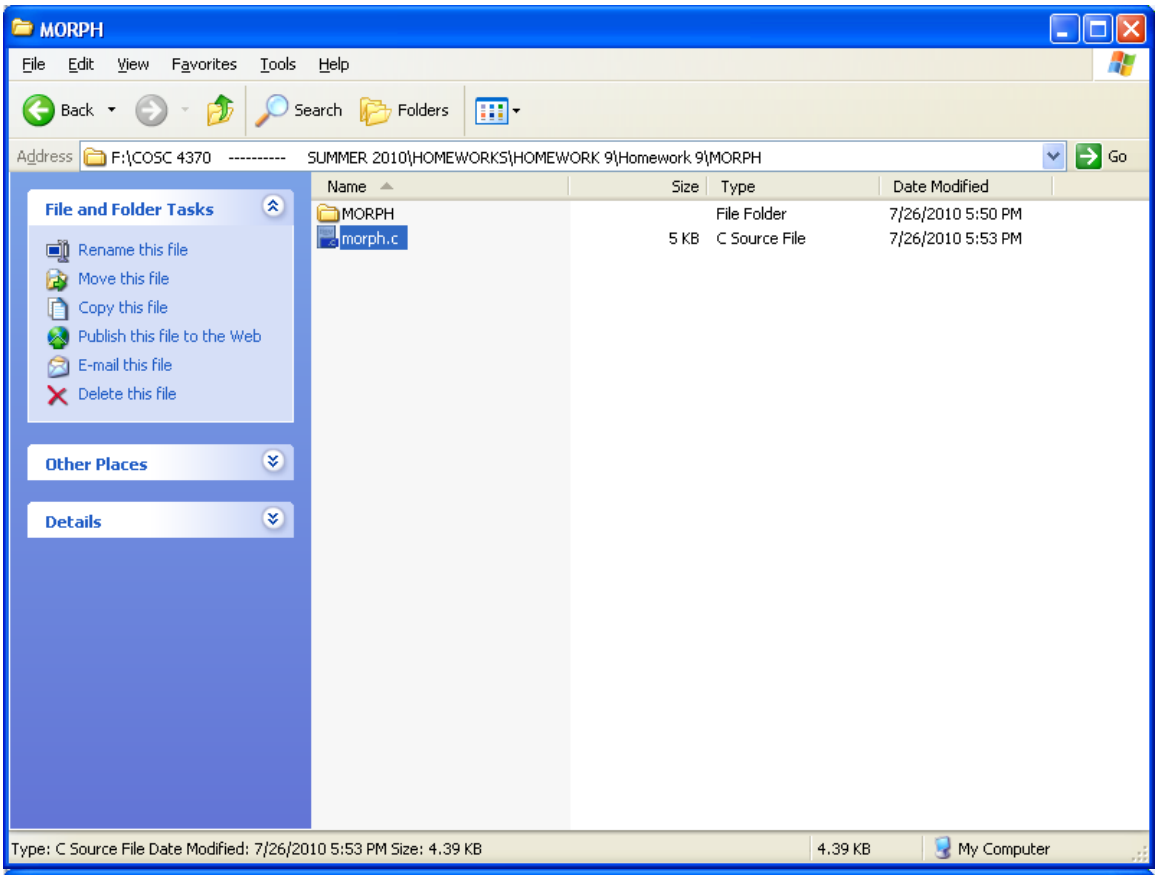

```
//coloredtexturedteapot.c

#include <GL/glew.h>
#include <stdlib.h>
#include <GL/glut.h>


static GLfloat theta[] = {0.0,0.0,0.0};
static GLint axis = 2;

    GLuint tex;

void display(void)
{

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        glBindTexture(GL_TEXTURE_CUBE_MAP, tex);

      glLoadIdentity();
      glRotatef(theta[0], 1.0, 0.0, 0.0);
```

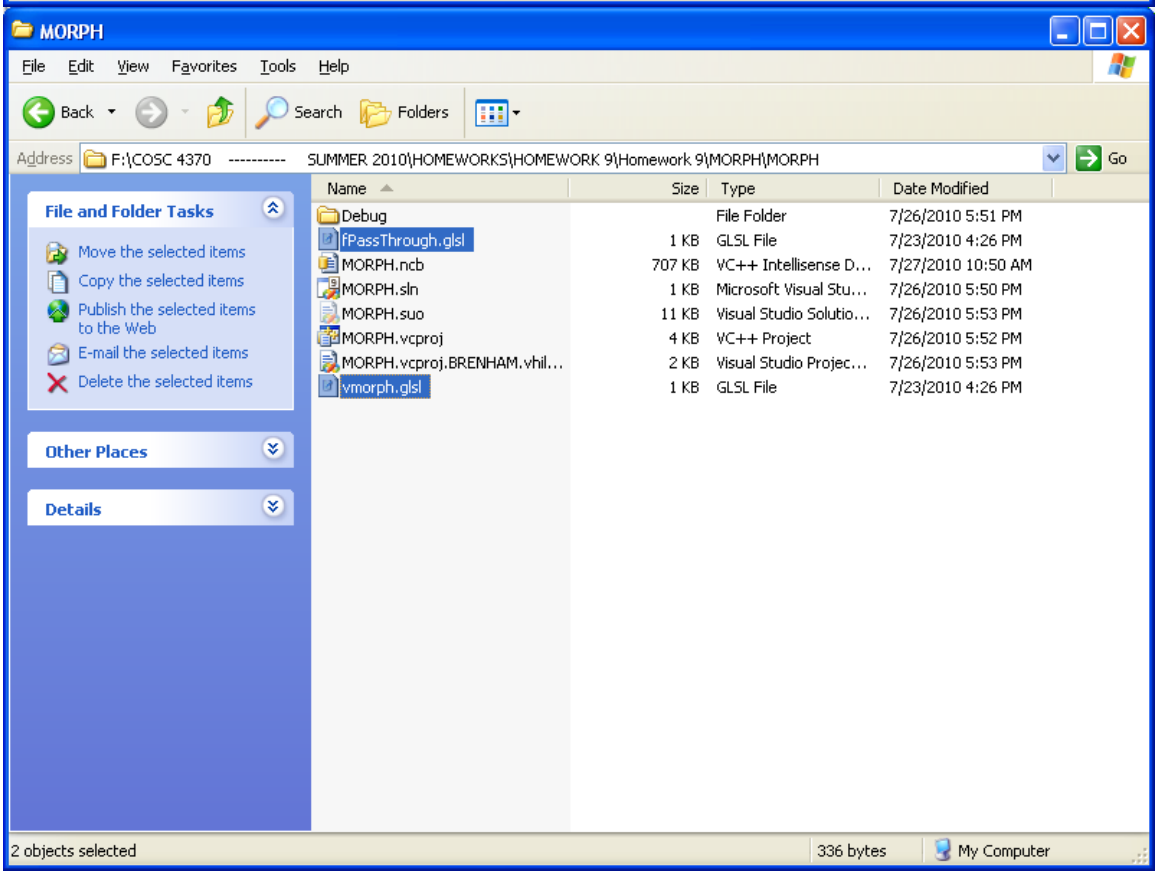
```
      glRotatef(theta[1], 0.0, 1.0, 0.0);
      glRotatef(theta[2], 0.0, 0.0, 1.0);

      glutSolidTeapot(1.0);


      glutSwapBuffers();
}

void spinCube()
{


      theta[axis] += 2.0;
      if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
      glutPostRedisplay();
}

void mouse(int btn, int state, int x, int y)
{

      if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
      if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
      if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN) axis = 2;
}

void myReshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        glOrtho(-2.0, 2.0, -2.0 * (GLfloat) h / (GLfloat) w,
            2.0 * (GLfloat) h / (GLfloat) w, -10.0, 10.0);
    else
        glOrtho(-2.0 * (GLfloat) w / (GLfloat) h,
            2.0 * (GLfloat) w / (GLfloat) h, -2.0, 2.0, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
}

void key(char k, int x, int y)
{
      if(k == '1') glutIdleFunc(spinCube);
      if(k == '2') glutIdleFunc(NULL);
        if(k == 'q') exit(0);
}

void init()
{

    GLubyte red[3] = {255, 0, 0};
    GLubyte green[3] = {0, 255, 0};
    GLubyte blue[3] = {0, 0, 255};
    GLubyte cyan[3] = {0, 255, 255};
    GLubyte magenta[3] = {255, 0, 255};
    GLubyte yellow[3] = {255, 255, 0};
```

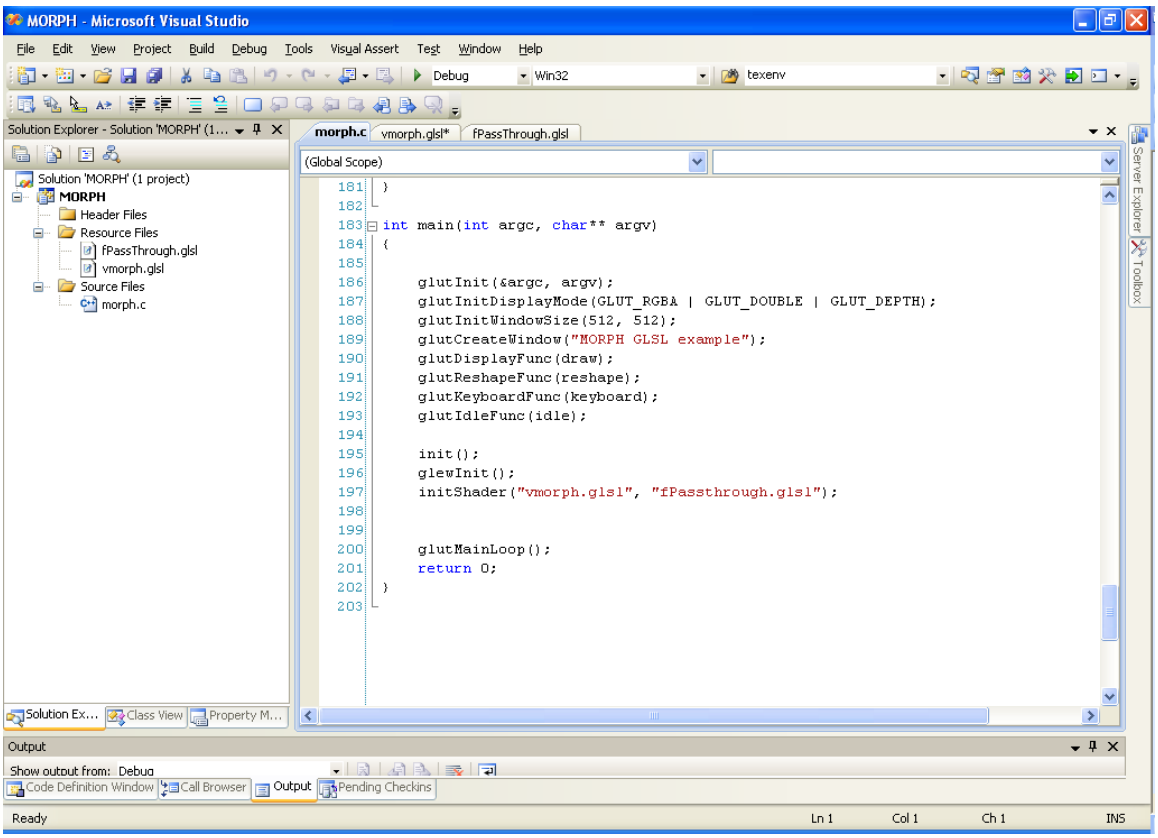
```
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_TEXTURE_GEN_S);
    glEnable(GL_TEXTURE_GEN_T);
    glEnable(GL_TEXTURE_GEN_R);
    glEnable(GL_TEXTURE_CUBE_MAP);

   glGenTextures(1, &tex);
   glBindTexture(GL_TEXTURE_CUBE_MAP, tex);

   glTexGeni(GL_S, GL_TEXTURE_GEN_MODE,GL_REFLECTION_MAP);
   glTexGeni(GL_T, GL_TEXTURE_GEN_MODE,GL_REFLECTION_MAP);
   glTexGeni(GL_R, GL_TEXTURE_GEN_MODE,GL_REFLECTION_MAP);
   glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X
,0,3,1,1,0,GL_RGB,GL_UNSIGNED_BYTE, red);
   glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X
,0,3,1,1,0,GL_RGB,GL_UNSIGNED_BYTE, green);
   glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y
,0,3,1,1,0,GL_RGB,GL_UNSIGNED_BYTE, blue);
   glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y
,0,3,1,1,0,GL_RGB,GL_UNSIGNED_BYTE, cyan);
   glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z
,0,3,1,1,0,GL_RGB,GL_UNSIGNED_BYTE, magenta);
   glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z
,0,3,1,1,0,GL_RGB,GL_UNSIGNED_BYTE, yellow);
   glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_WRAP_S,GL_REPEAT);
   glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_WRAP_T,GL_REPEAT);
   glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_WRAP_R,GL_REPEAT);

glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_MAG_FILTER,GL_NEAREST);

glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_MIN_FILTER,GL_NEAREST);

   glClearColor(1.0,1.0,1.0,1.0);
}

void
main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("COLORED TEXTURED TEAPOT non GLSL Example");

    init();

    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutIdleFunc(spinCube);
    glutIdleFunc(NULL);
    glutMouseFunc(mouse);

    glutKeyboardFunc(key);
    glutMainLoop();
}
```

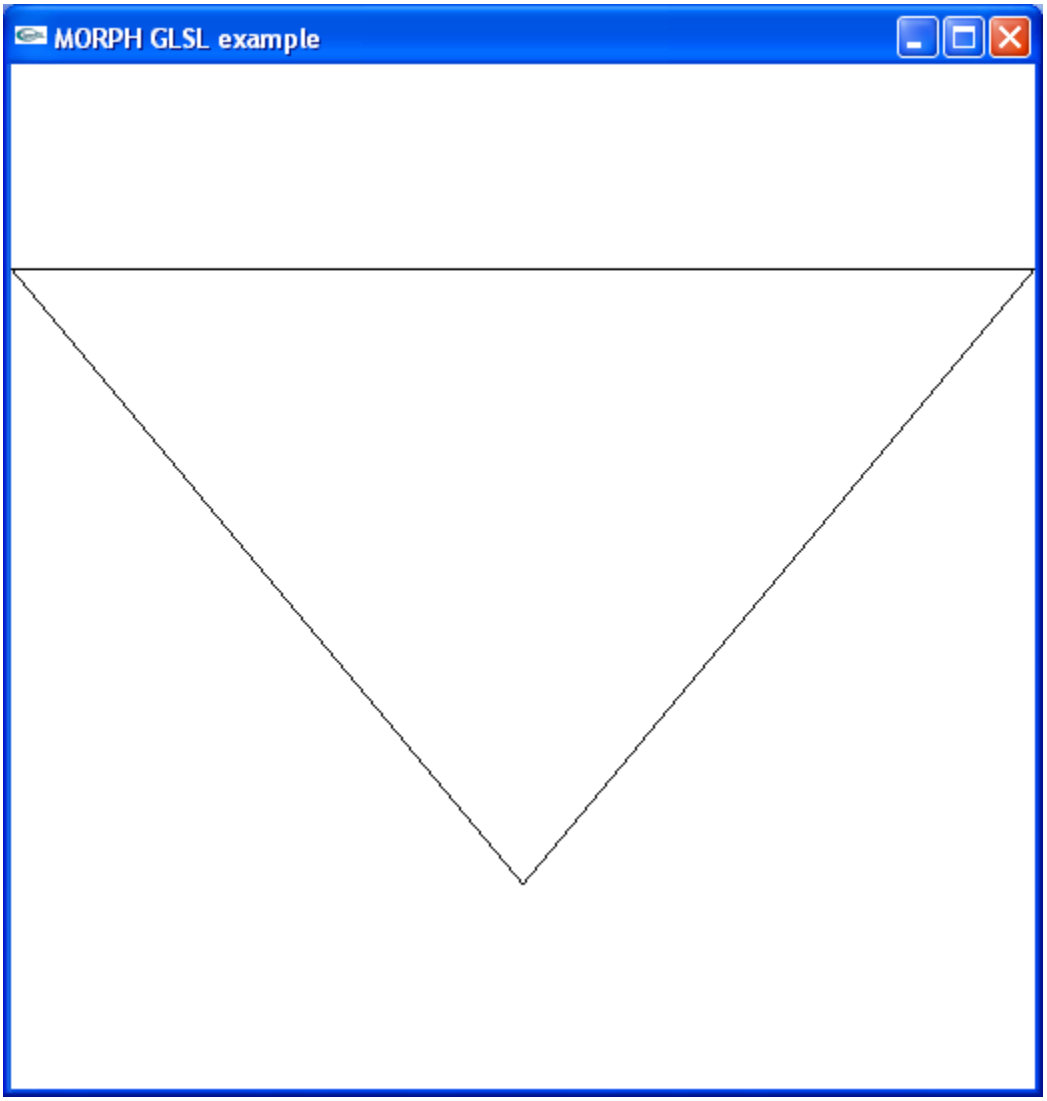
3. MORPH


Homework 9
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9
Go
File and Folder Tasks
Rename this folder
Move this folder
Copy this folder
Publish this folder to the Web
Share this folder
E-mail this folder's files
Delete this folder
Other Places
Details
Name Size Type Date Modified
BUMPMAP File Folder 7/25/2010 4:30 PM
COLORED TEXTERED TEAPOT File Folder 7/25/2010 4:30 PM
MORPH File Folder 7/25/2010 4:30 PM
NONPHOTO File Folder 7/25/2010 4:30 PM
PARTICLE File Folder 7/25/2010 4:30 PM
PHONG File Folder 7/25/2010 4:30 PM
READ ME File Folder 7/25/2010 4:30 PM
WAVE File Folder 7/25/2010 4:30 PM
WAVE2 File Folder 7/25/2010 4:30 PM
1 objects selected
My Computer


MORPH
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\MORPH
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places
Details
Name Size Type Date Modified
MORPH File Folder 7/26/2010 5:50 PM
morph.c 5 KB C Source File 7/26/2010 5:53 PM
Type: C Source File Date Modified: 7/26/2010 5:53 PM Size: 4.39 KB
4.39 KB
My Computer


MORPH
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\MORPH\MORPH
Go
File and Folder Tasks
Move the selected items
Copy the selected items
Publish the selected items to the Web
E-mail the selected items
Delete the selected items
Other Places
Details
Name Size Type Date Modified
Debug File Folder 7/26/2010 5:51 PM
fPassThrough.glsl 1 KB GLSL File 7/23/2010 4:26 PM
MORPH.ncb 707 KB VC++ Intellisense D... 7/27/2010 10:50 AM
MORPH.sln 1 KB Microsoft Visual Stu... 7/26/2010 5:50 PM
MORPH.suo 11 KB Visual Studio Solutio... 7/26/2010 5:53 PM
MORPH.vcproj 4 KB VC++ Project 7/26/2010 5:52 PM
MORPH.vcproj.BRENHAM.vhil... 2 KB Visual Studio Projec... 7/26/2010 5:53 PM
vmorph.glsl 1 KB GLSL File 7/23/2010 4:26 PM
2 objects selected
336 bytes
My Computer


MORPH - Microsoft Visual Studio
File Edit View Project Build Debug Tools Visual Assert Test Window Help
Debug
Win32
texenv
Solution Explorer - Solution 'MORPH' (1...
Solution 'MORPH' (1 project)
MORPH
Header Files
Resource Files
fPassThrough.glsl
vmorph.glsl
Source Files
morph.c
morph.c
vmorph.glsl*
fPassThrough.glsl
(Global Scope)
181 }
182
183 int main(int argc, char** argv)
184 {
185
186 glutInit(&argc, argv);
187 glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
188 glutInitWindowSize(512, 512);
189 glutCreateWindow("MORPH GLSL example");
190 glutDisplayFunc(draw);
191 glutReshapeFunc(reshape);
192 glutKeyboardFunc(keyboard);
193 glutIdleFunc(idle);
194
195 init();
196 glewInit();
197 initShader("vmorph.glsl", "fPassthrough.glsl");
198
199
200 glutMainLoop();
201 return 0;
202 }
203
Server Explorer
Toolbox
Solution Ex...
Class View
Property M...
Output
Show output from: Debug
Code Definition Window
Call Browser
Output
Pending Checkins
Ready
Ln 1
Col 1
Ch 1
INS




```
// morph.c

/* sets up flat mesh */
/* sets up elapsed time parameter for use by shaders */

#include <GL/glew.h>

#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <GL/glut.h>

GLuint          program;
GLint           timeParam;
GLint           vertices_two_location;

const GLfloat vertices_one[3][2] = {{0.0, 0.0},{0.5,1.0},{1.0, 0.0}};
const GLfloat vertices_two[3][2] = {{0.0, 1.0},{0.5,0.0},{1.0, 1.0}};

/* shader reader */
/* creates null terminated string from file */
```

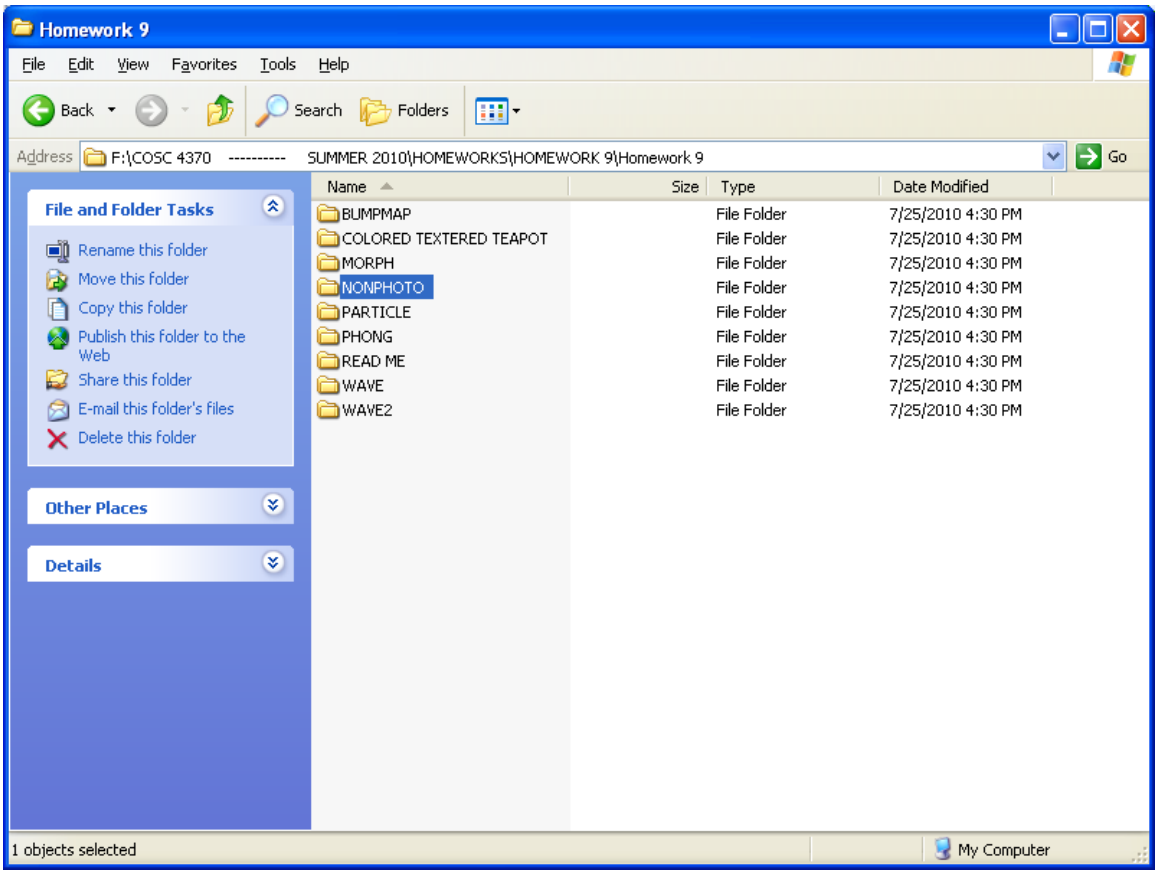
```
static char* readShaderSource(const char* shaderFile)
{
    FILE* fp = fopen(shaderFile, "rb");

    char* buf;
    long size;

    if(fp==NULL) return NULL;
    fseek(fp, 0L, SEEK_END);
    size = ftell(fp);
    fseek(fp, 0L, SEEK_SET);
    buf = (char*) malloc((size+1) * sizeof(char));
    fread(buf, 1, size, fp);
      buf[size] = '\0';
    fclose(fp);
    return buf;
}

/* error printing function */

static void checkError(GLint status, const char *msg)
{
    if (status != GL_TRUE)
    {
        printf("%s\n", msg);
        exit(EXIT_FAILURE);
    }
}

/* standard OpenGL initialization */

static void init()
{

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glColor3f(0.0,0.0,0.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,1.0,0.0,1.0);

    glEnable(GL_DEPTH_TEST);
}

/* GLSL initialization */

static void initShader(const GLchar* vShaderFile, const GLchar*
fShaderFile)
{
    GLint status;
    GLchar *vSource, *fSource;
    GLuint vShader, fShader;

    /* read shader files */

    vSource = readShaderSource(vShaderFile);
```

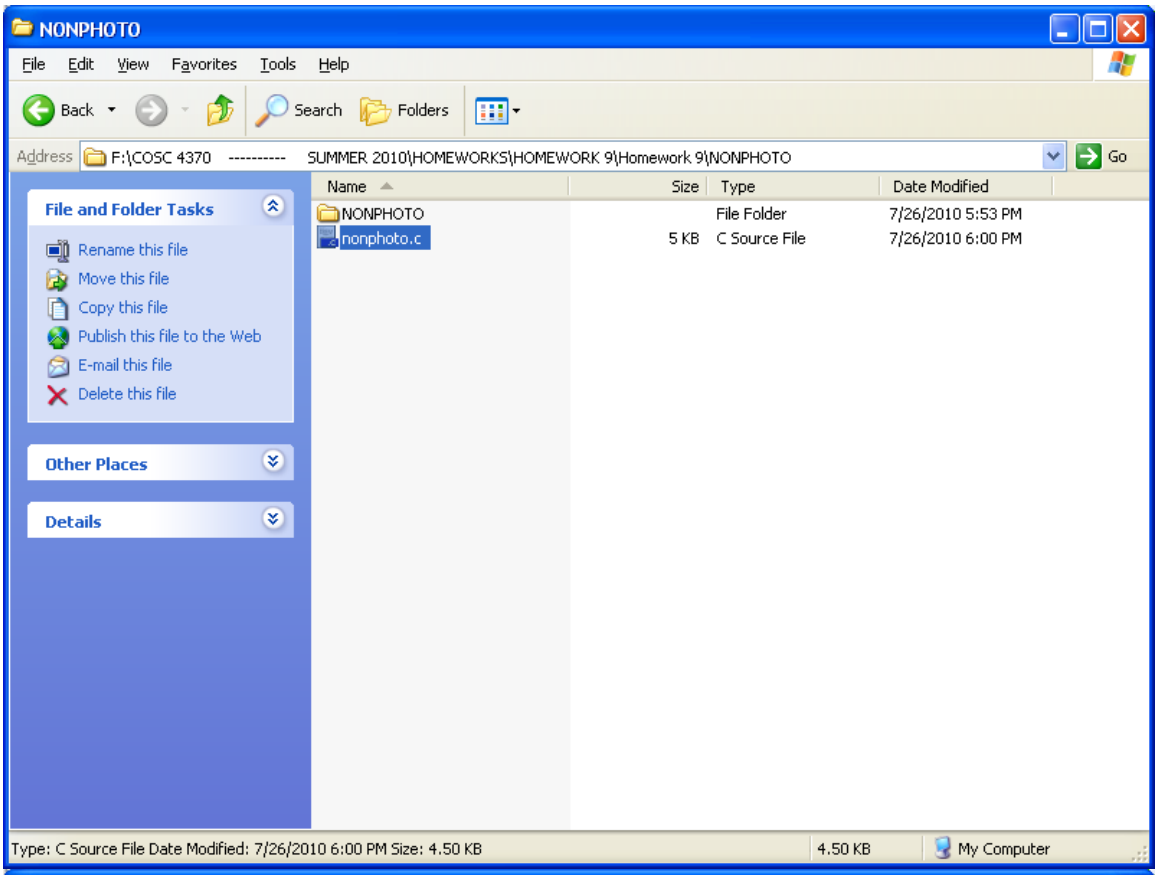
```
    if(vSource==NULL)
    {
        printf( "Failed to read vertex shaderi\n");
        exit(EXIT_FAILURE);
    }

    fSource = readShaderSource(fShaderFile);
    if(fSource==NULL)
    {
        printf("Failed to read fragment shader");
        exit(EXIT_FAILURE);
    }

    /* create program and shader objects */

    vShader = glCreateShader(GL_VERTEX_SHADER);
    fShader = glCreateShader(GL_FRAGMENT_SHADER);
    program = glCreateProgram();

    /* attach shaders to the program object */

    glAttachShader(program, vShader);
    glAttachShader(program, fShader);

    /* read shaders */

    glShaderSource(vShader, 1, (const GLchar**) &vSource, NULL);
    glShaderSource(fShader, 1, (const GLchar**) &fSource, NULL);

    /* compile shaders */

    glCompileShader(vShader);
    glCompileShader(fShader);

    /* error check */

    glGetShaderiv(vShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the vertex shader.");

    glGetShaderiv(fShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the fragment shader.");

    /* link */

    glLinkProgram(program);
    glGetProgramiv(program, GL_LINK_STATUS, &status);
    checkError(status, "Failed to link the shader program object.");

    /* use program object */

    glUseProgram(program);

    /* set up uniform parameter */

    timeParam = glGetUniformLocation(program, "time");
    vertices_two_location = glGetAttribLocation(program, "vertices2");
}
```

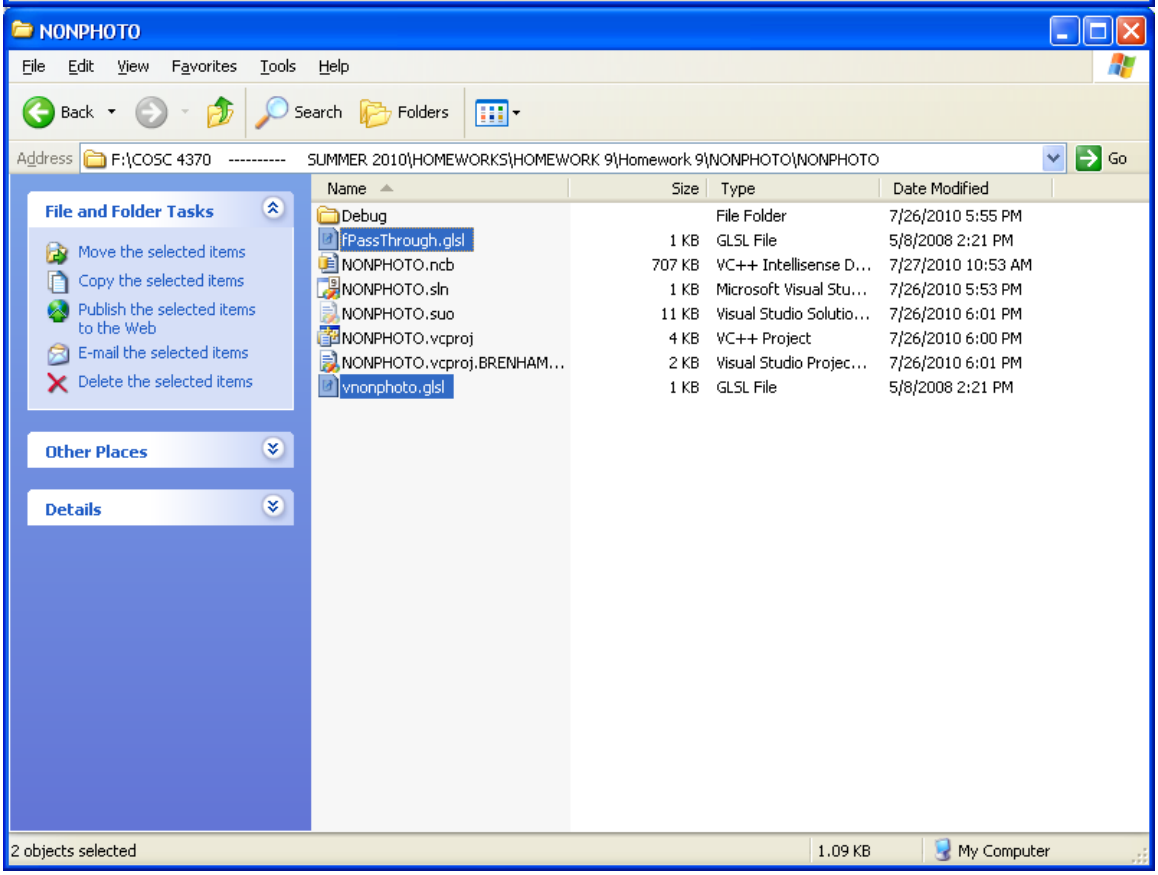
```
static void draw(void)
{
    /* send elapsed time to shaders */

    glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glBegin(GL_LINE_LOOP);
       glVertexAttrib2fv(vertices_two_location, &vertices_two[0][0]);
       glVertex2fv(vertices_one[0]);
       glVertexAttrib2fv(vertices_two_location, &vertices_two[1][0]);
       glVertex2fv(vertices_one[1]);
       glVertexAttrib2fv(vertices_two_location, &vertices_two[2][0]);
       glVertex2fv(vertices_one[2]);
    glEnd();

    glutSwapBuffers();
}

static void reshape(int w, int h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,1.0,0.0,1.0);

    glViewport(0, 0, w, h);
}

static void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 27:
    case 'Q':
    case 'q':
        exit(EXIT_SUCCESS);
        break;
    default:
        break;
    }
}

void idle()
{
   glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));
   glutPostRedisplay();
}

int main(int argc, char** argv)
{

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(512, 512);
    glutCreateWindow("MORPH GLSL example");
    glutDisplayFunc(draw);
    glutReshapeFunc(reshape);
```

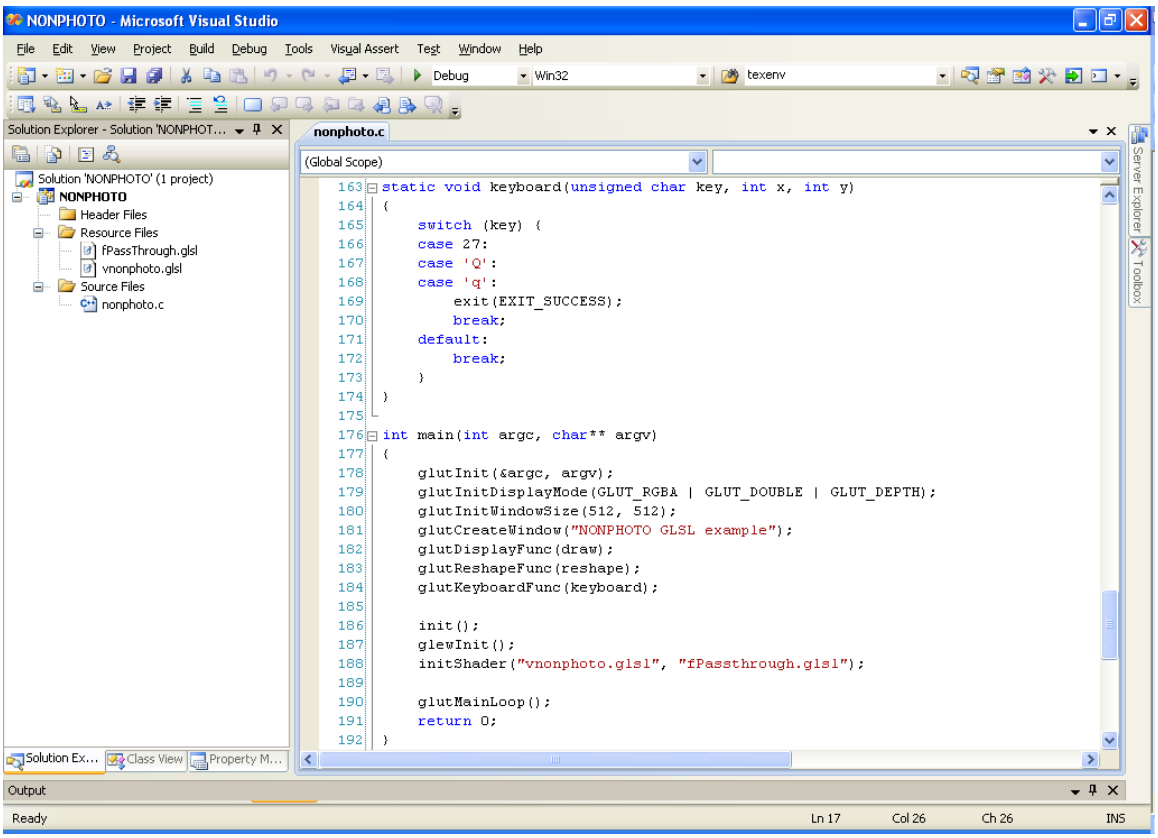
```
    glutKeyboardFunc(keyboard);
    glutIdleFunc(idle);

    init();
      glewInit();
    initShader("vmorph.glsl", "fPassthrough.glsl");


    glutMainLoop();
    return 0;
}


// vmorph.glsl

attribute vec4 vertices2;
uniform float time;

void main()
{
    float s =  0.5*(1.0+sin(0.001*time));
    vec4 t = mix(gl_Vertex, vertices2, s);
    gl_Position = gl_ModelViewProjectionMatrix*t;
    gl_FrontColor =  gl_Color;
}

// fPassThrough.glsl
// Pass through fragment shader.

void main()
{
    gl_FragColor = gl_Color;
}
```

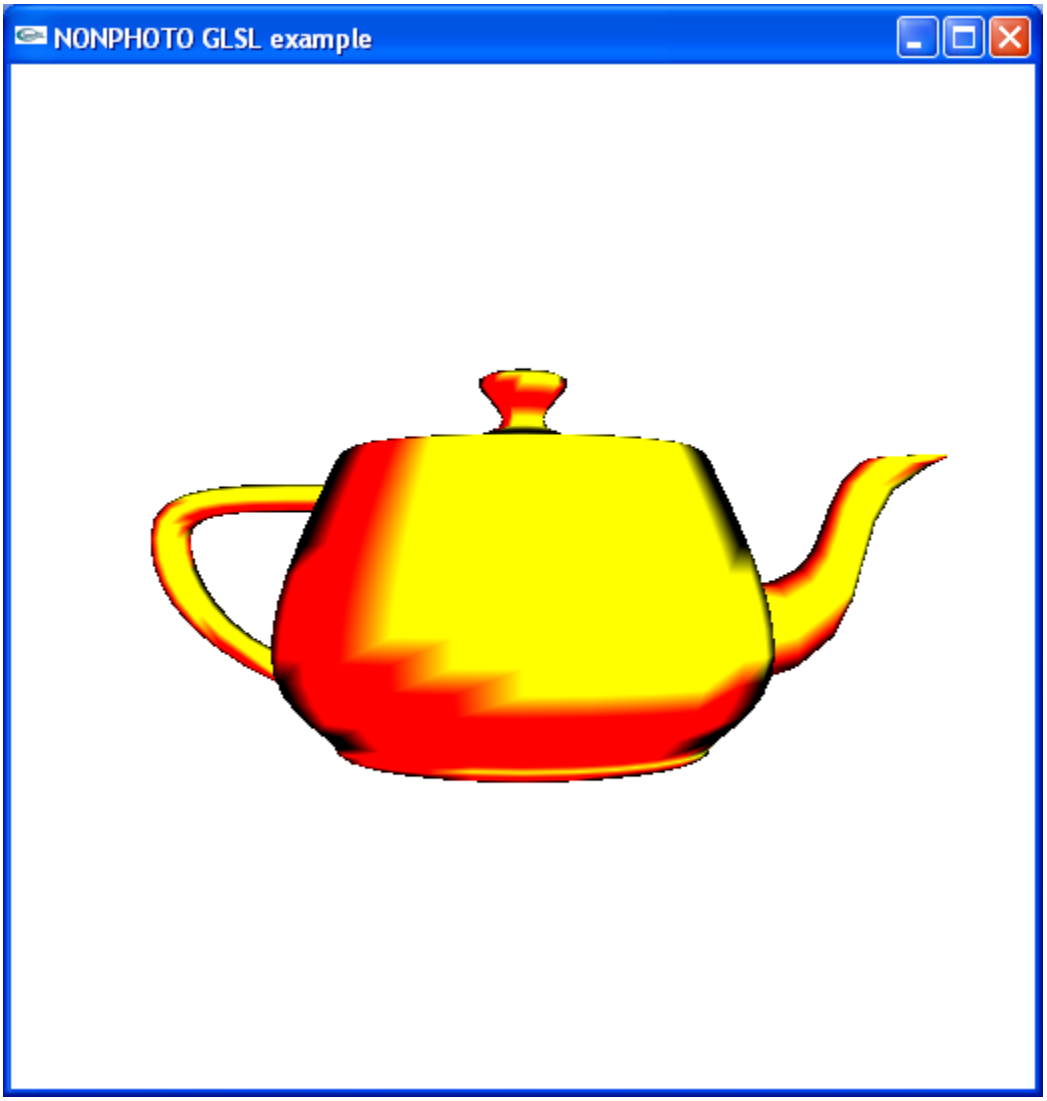
4. NONPHOTO


Homework 9
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9
Go
File and Folder Tasks
Rename this folder
Move this folder
Copy this folder
Publish this folder to the Web
Share this folder
E-mail this folder's files
Delete this folder
Other Places
Details
Name Size Type Date Modified
BUMPMAP File Folder 7/25/2010 4:30 PM
COLORED TEXTERED TEAPOT File Folder 7/25/2010 4:30 PM
MORPH File Folder 7/25/2010 4:30 PM
NONPHOTO File Folder 7/25/2010 4:30 PM
PARTICLE File Folder 7/25/2010 4:30 PM
PHONG File Folder 7/25/2010 4:30 PM
READ ME File Folder 7/25/2010 4:30 PM
WAVE File Folder 7/25/2010 4:30 PM
WAVE2 File Folder 7/25/2010 4:30 PM
1 objects selected
My Computer


NONPHOTO
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\NONPHOTO
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places
Details
Name Size Type Date Modified
NONPHOTO File Folder 7/26/2010 5:53 PM
nonphoto.c 5 KB C Source File 7/26/2010 6:00 PM
Type: C Source File Date Modified: 7/26/2010 6:00 PM Size: 4.50 KB
4.50 KB
My Computer


NONPHOTO
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\NONPHOTO\NONPHOTO
Go
File and Folder Tasks
Move the selected items
Copy the selected items
Publish the selected items to the Web
E-mail the selected items
Delete the selected items
Other Places
Details
Name Size Type Date Modified
Debug File Folder 7/26/2010 5:55 PM
fPassThrough.glsl 1 KB GLSL File 5/8/2008 2:21 PM
NONPHOTO.ncb 707 KB VC++ Intellisense D... 7/27/2010 10:53 AM
NONPHOTO.sln 1 KB Microsoft Visual Stu... 7/26/2010 5:53 PM
NONPHOTO.suo 11 KB Visual Studio Solutio... 7/26/2010 6:01 PM
NONPHOTO.vcproj 4 KB VC++ Project 7/26/2010 6:00 PM
NONPHOTO.vcproj.BRENHAM... 2 KB Visual Studio Projec... 7/26/2010 6:01 PM
vnonphoto.glsl 1 KB GLSL File 5/8/2008 2:21 PM
2 objects selected
1.09 KB
My Computer


NONPHOTO - Microsoft Visual Studio
File Edit View Project Build Debug Tools Visual Assert Test Window Help
Debug
Win32
texenv
Solution Explorer - Solution 'NONPHOT...
Solution 'NONPHOTO' (1 project)
NONPHOTO
Header Files
Resource Files
fPassThrough.glsl
vnonphoto.glsl
Source Files
nonphoto.c
Solution Ex...
Class View
Property M...
nonphoto.c
(Global Scope)
163 static void keyboard(unsigned char key, int x, int y)
164 {
165 switch (key) {
166 case 27:
167 case 'Q':
168 case 'q':
169 exit(EXIT_SUCCESS);
170 break;
171 default:
172 break;
173 }
174 }
175
176 int main(int argc, char** argv)
177 {
178 glutInit(&argc, argv);
179 glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
180 glutInitWindowSize(512, 512);
181 glutCreateWindow("NONPHOTO GLSL example");
182 glutDisplayFunc(draw);
183 glutReshapeFunc(reshape);
184 glutKeyboardFunc(keyboard);
185
186 init();
187 glewInit();
188 initShader("vnonphoto.glsl", "fPassthrough.glsl");
189
190 glutMainLoop();
191 return 0;
192 }
Server Explorer
Toolbox
Output
Ready
Ln 17
Col 26
Ch 26
INS




```
// nonphoto.c

/* display teapot with vertex and fragment shaders */
/* sets up elapsed time parameter for use by shaders */

#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <GL/glew.h>
#include <GL/glut.h>


const GLdouble nearVal     = 1.0;
const GLdouble farVal      = 20.0;
const GLfloat  lightPos[4] = {3.0f, 3.0f, 3.0f, 1.0f};
GLuint         program     = 0;
GLint          timeParam;

/* shader reader */
```

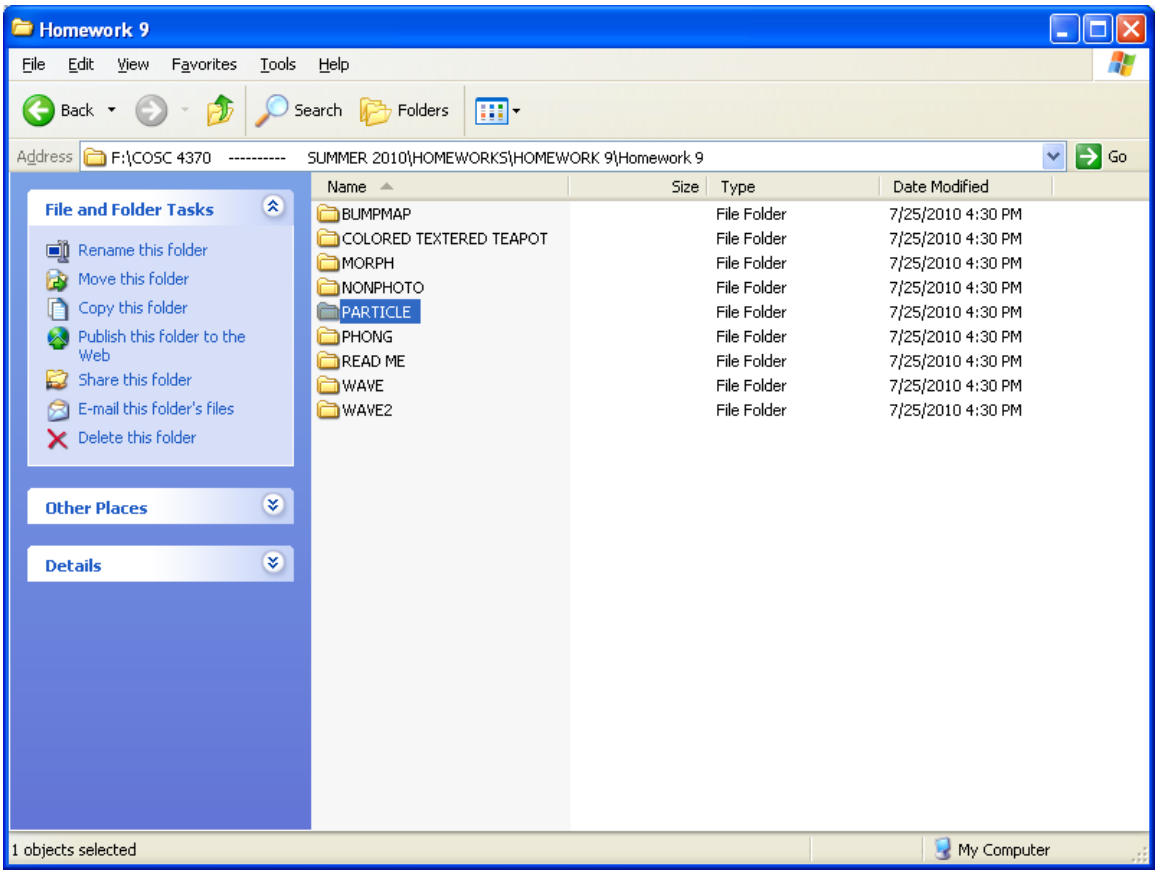
```
/* creates null terminated string from file */

static char* readShaderSource(const char* shaderFile)
{
    FILE* fp = fopen(shaderFile, "rb");

    char* buf;
    long size;

    if(fp==NULL) return NULL;
    fseek(fp, 0L, SEEK_END);
    size = ftell(fp);
    fseek(fp, 0L, SEEK_SET);
    buf = (char*) malloc((size+1) * sizeof(char));
    fread(buf, 1, size, fp);
      buf[size] = '\0';
    fclose(fp);
    return buf;
}

/* error printing function */

static void checkError(GLint status, const char *msg)
{
    if (!status)
    {
        printf("%s\n", msg);
        exit(EXIT_FAILURE);
    }
}

/* standard OpenGL initialization */

static void init()
{
    const float teapotColor[]     = {0.3f, 0.5f, 0.4f, 1.0f};
    const float teapotSpecular[]  = {0.8f, 0.8f, 0.8f, 1.0f};
    const float teapotShininess[] = {80.0f};

    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, teapotColor);
    glMaterialfv(GL_FRONT, GL_SPECULAR, teapotSpecular);
    glMaterialfv(GL_FRONT, GL_SHININESS, teapotShininess);

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double) glutGet(GLUT_WINDOW_WIDTH) / (double)
glutGet(GLUT_WINDOW_HEIGHT), nearVal, farVal);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
}
```

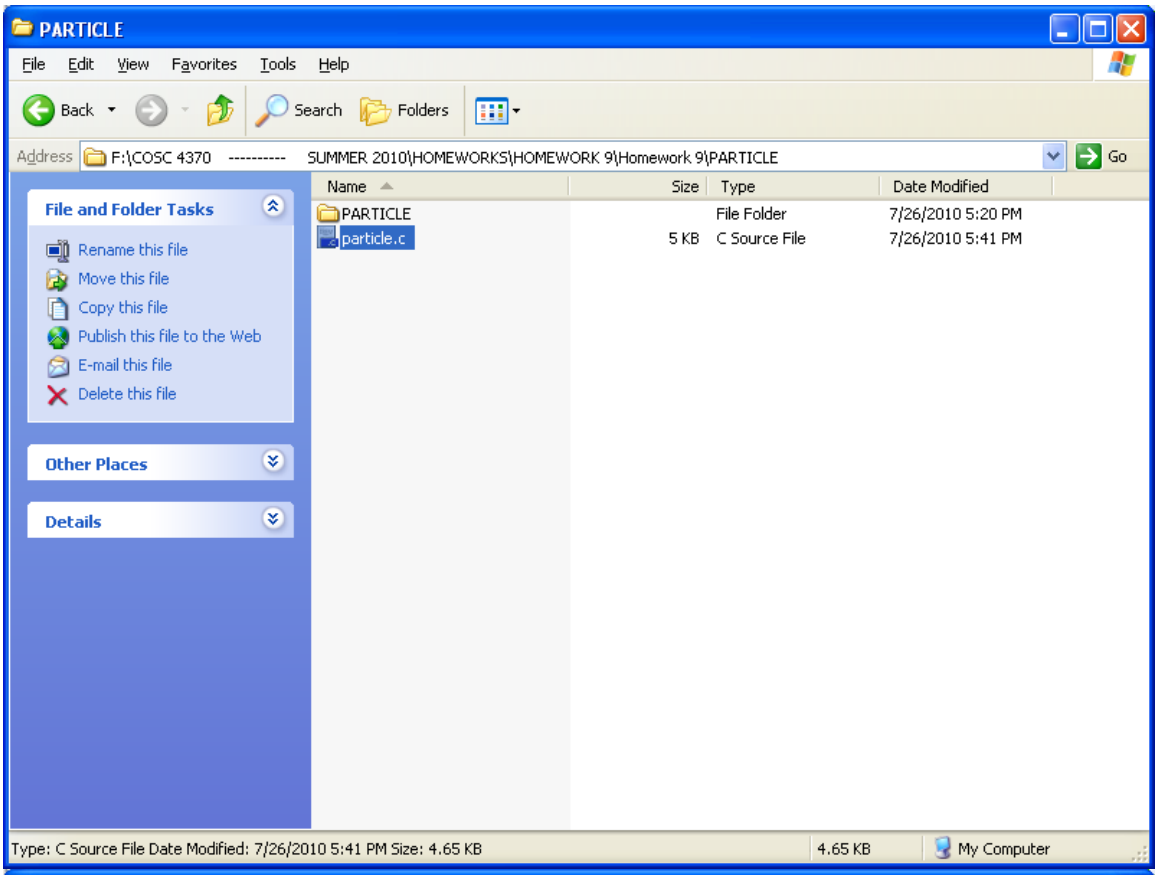
```
/* GLSL initialization */

static void initShader(const GLchar* vShaderFile, const GLchar*
fShaderFile)
{
    GLint status = glGetError()==GL_NO_ERROR;
    GLchar *vSource, *fSource;
    GLuint vShader, fShader;

    /* read shader files */

    vSource = readShaderSource(vShaderFile);
    checkError(status, "Failed to read vertex shader");

    fSource = readShaderSource(fShaderFile);
    checkError(status, "Failed to read fragment shader");

    /* create program and shader objects */

    vShader = glCreateShader(GL_VERTEX_SHADER);
    fShader = glCreateShader(GL_FRAGMENT_SHADER);
    program = glCreateProgram();

    /* attach shaders to the program object */

    glAttachShader(program, vShader);
    glAttachShader(program, fShader);

    /* read shaders */

    glShaderSource(vShader, 1, (const GLchar**) &vSource, NULL);
    glShaderSource(fShader, 1, (const GLchar**) &fSource, NULL);

    /* compile shaders */

    glCompileShader(vShader);
    glCompileShader(fShader);

    /* error check */

    glGetShaderiv(vShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the vertex shader.");

    glGetShaderiv(fShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the fragment shader.");

    /* link */

    glLinkProgram(program);
    glGetProgramiv(program, GL_LINK_STATUS, &status);
    checkError(status, "Failed to link the shader program object.");

    /* use program object */

    glUseProgram(program);
```

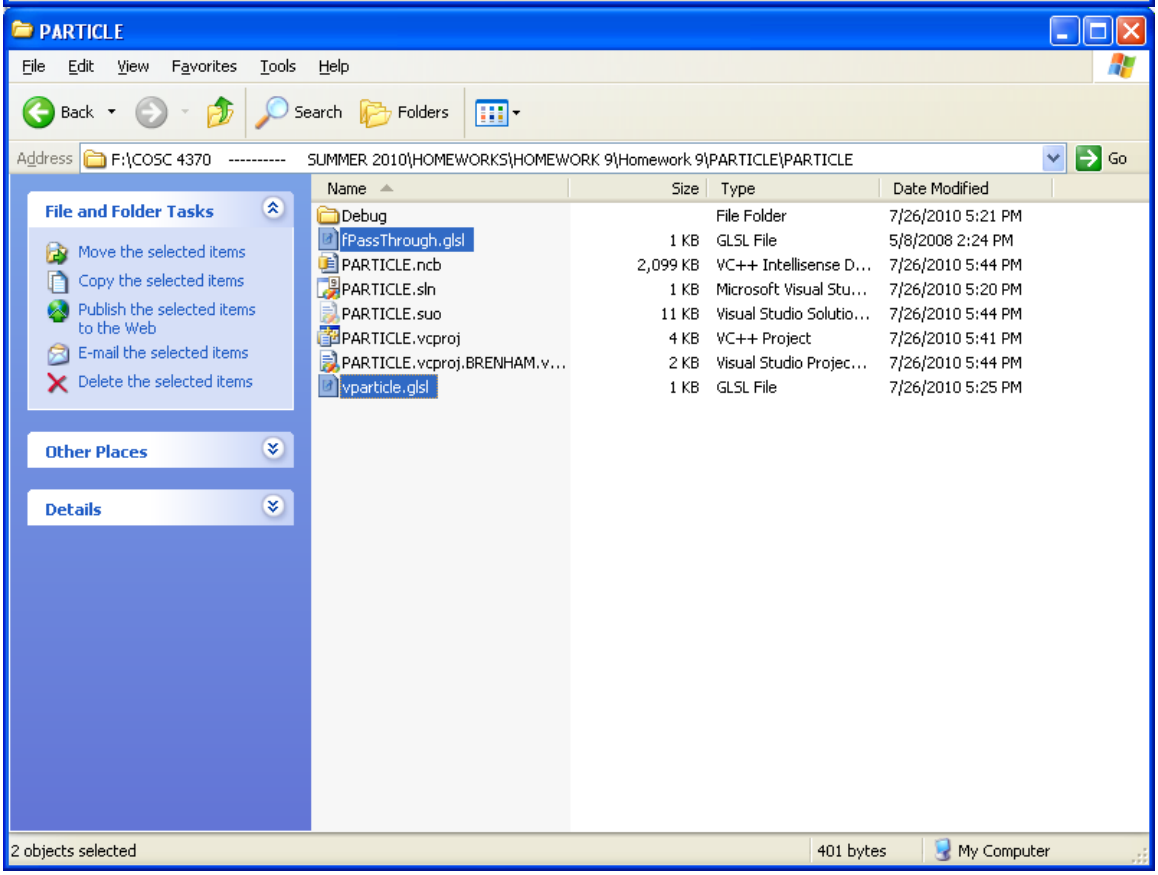
```
    /* set up uniform parameter */

    timeParam = glGetUniformLocation(program, "time");
}

static void draw(void)
{
    /* send elapsed time to shaders */

    glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glTranslatef(0.0f, 0.0f, -5.0f);
    glutSolidTeapot(1.0);
    glPopMatrix();
    glutSwapBuffers();
}

static void reshape(int w, int h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double) w / (double) h, nearVal, farVal);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glViewport(0, 0, w, h);

    glutPostRedisplay();
}

static void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 27:
    case 'Q':
    case 'q':
        exit(EXIT_SUCCESS);
        break;
    default:
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(512, 512);
    glutCreateWindow("NONPHOTO GLSL example");
    glutDisplayFunc(draw);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);

    init();
      glewInit();
    initShader("vnonphoto.glsl", "fPassthrough.glsl");
```

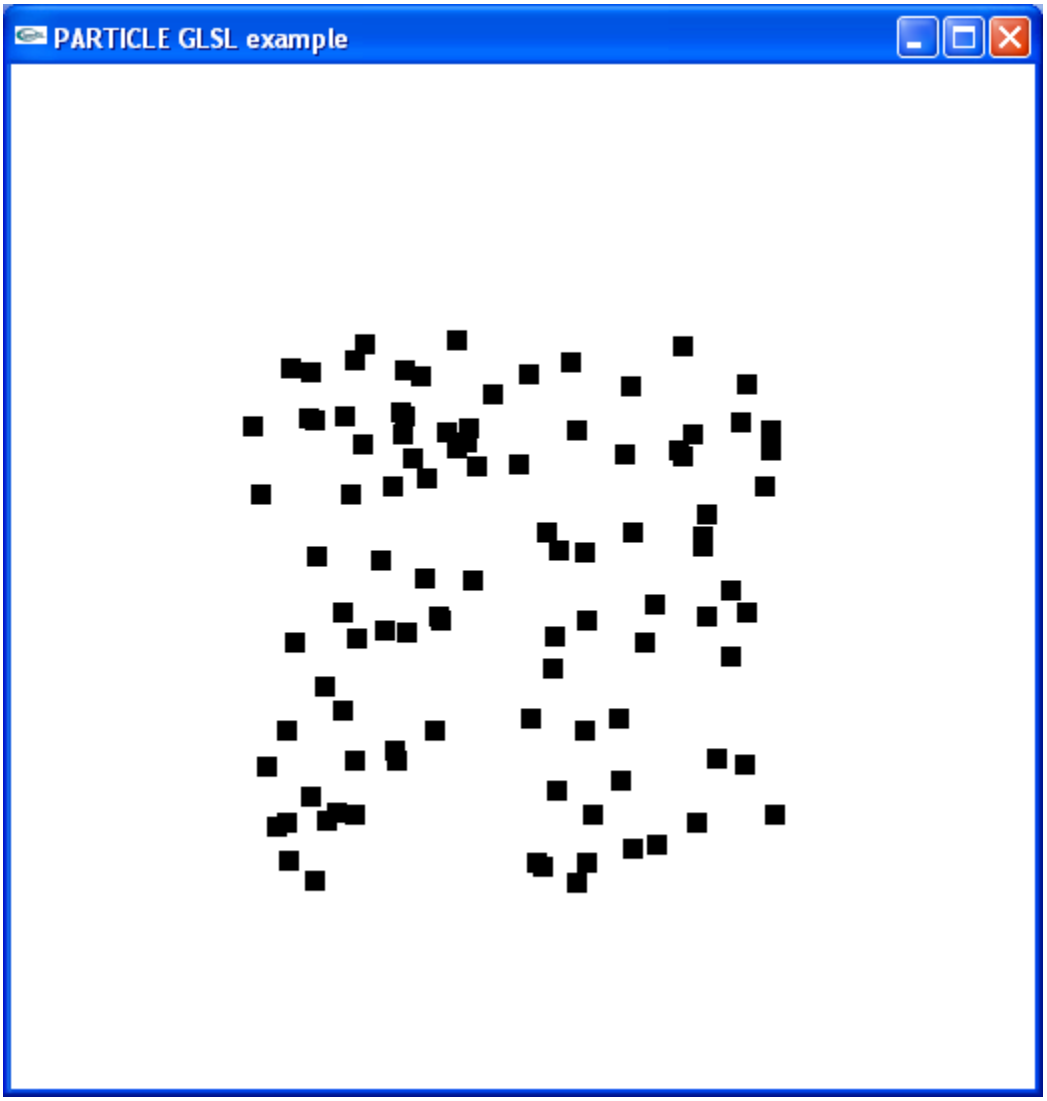
```
    glutMainLoop();
    return 0;
}
// vnonphoto.glsl

void main()
{
    const vec4 yellow = vec4(1.0, 1.0, 0.0, 1.0);
    const vec4 red = vec4(1.0, 0.0, 0.0, 1.0);
    const vec4 black = vec4(0.0, 0.0, 0.0, 1.0);

    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;

    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    vec4 eyePosition = gl_ModelViewMatrix * gl_Vertex;
    vec4 eyeLightPos = gl_LightSource[0].position;
    vec3 N = normalize(gl_NormalMatrix * gl_Normal);
    vec3 L = normalize(eyeLightPos.xyz - eyePosition.xyz);
    vec3 E = -normalize(eyePosition.xyz);
    vec3 H = normalize(L + E);
    float Kd = max(dot(L, N), 0.0);
    float Ks = pow(max(dot(N, H), 0.0), gl_FrontMaterial.shininess);
    float Ka = 0.0;

    ambient = Ka*gl_FrontLightProduct[0].ambient;
    diffuse = Kd*gl_FrontLightProduct[0].diffuse;
    specular = Ks*gl_FrontLightProduct[0].specular;
    if(Kd > 0.6) diffuse = yellow;
      else diffuse = red;
    gl_FrontColor = diffuse;
    if(abs(dot(E,N))<0.25) gl_FrontColor = black;
}


// fPassThrough.glsl

// Pass through fragment shader.

void main()
{
    gl_FragColor = gl_Color;
}
```

5. PARTICLE


Homework 9
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9
Go
File and Folder Tasks
Rename this folder
Move this folder
Copy this folder
Publish this folder to the Web
Share this folder
E-mail this folder's files
Delete this folder
Other Places
Details
Name Size Type Date Modified
BUMPMAP File Folder 7/25/2010 4:30 PM
COLORED TEXTERED TEAPOT File Folder 7/25/2010 4:30 PM
MORPH File Folder 7/25/2010 4:30 PM
NONPHOTO File Folder 7/25/2010 4:30 PM
PARTICLE File Folder 7/25/2010 4:30 PM
PHONG File Folder 7/25/2010 4:30 PM
READ ME File Folder 7/25/2010 4:30 PM
WAVE File Folder 7/25/2010 4:30 PM
WAVE2 File Folder 7/25/2010 4:30 PM
1 objects selected
My Computer


PARTICLE
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\PARTICLE
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places
Details
Name Size Type Date Modified
PARTICLE File Folder 7/26/2010 5:20 PM
particle.c 5 KB C Source File 7/26/2010 5:41 PM
Type: C Source File Date Modified: 7/26/2010 5:41 PM Size: 4.65 KB
4.65 KB
My Computer


PARTICLE
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\PARTICLE\PARTICLE
Go
File and Folder Tasks
Move the selected items
Copy the selected items
Publish the selected items to the Web
E-mail the selected items
Delete the selected items
Other Places
Details
Name Size Type Date Modified
Debug File Folder 7/26/2010 5:21 PM
fPassThrough.glsl 1 KB GLSL File 5/8/2008 2:24 PM
PARTICLE.ncb 2,099 KB VC++ Intellisense D... 7/26/2010 5:44 PM
PARTICLE.sln 1 KB Microsoft Visual Stu... 7/26/2010 5:20 PM
PARTICLE.suo 11 KB Visual Studio Solutio... 7/26/2010 5:44 PM
PARTICLE.vcproj 4 KB VC++ Project 7/26/2010 5:41 PM
PARTICLE.vcproj.BRENHAM.v... 2 KB Visual Studio Projec... 7/26/2010 5:44 PM
vparticle.glsl 1 KB GLSL File 7/26/2010 5:25 PM
2 objects selected
401 bytes
My Computer




```
//particle.c

/* sets up flat mesh */
/* sets up elapsed time parameter for use by shaders */

#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <GL/glew.h>
#include <GL/glut.h>
#include <math.h>


#define N 100

GLfloat velocity[N][2];

GLuint          program;
```

```
GLuint       vxParam, vyParam, timeParam;

GLchar *ebuffer; /* buffer for error messages */
GLsizei elength; /* length of error message */


/* shader reader */
/* creates null terminated string from file */

static char* readShaderSource(const char* shaderFile)
{
    FILE* fp = fopen(shaderFile, "rb");

    char* buf;
    long size;

    if(fp==NULL) return NULL;
    fseek(fp, 0L, SEEK_END);
    size = ftell(fp);
    fseek(fp, 0L, SEEK_SET);
    buf = (char*) malloc((size+1) * sizeof(char));
    fread(buf, 1, size, fp);
      buf[size] = '\0';
    fclose(fp);
    return buf;
}

/* error printing function */

static void checkError(GLint status, const char *msg)
{
    if (status != GL_TRUE)
    {
        printf("%s\n", msg);
        exit(EXIT_FAILURE);
    }
}

/* standard OpenGL initialization */

static void init()
{

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glColor3f(0.0,0.0,0.0);
    glPointSize(10.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,1.0,0.0,1.0);

    glEnable(GL_DEPTH_TEST);
}

/* GLSL initialization */
```

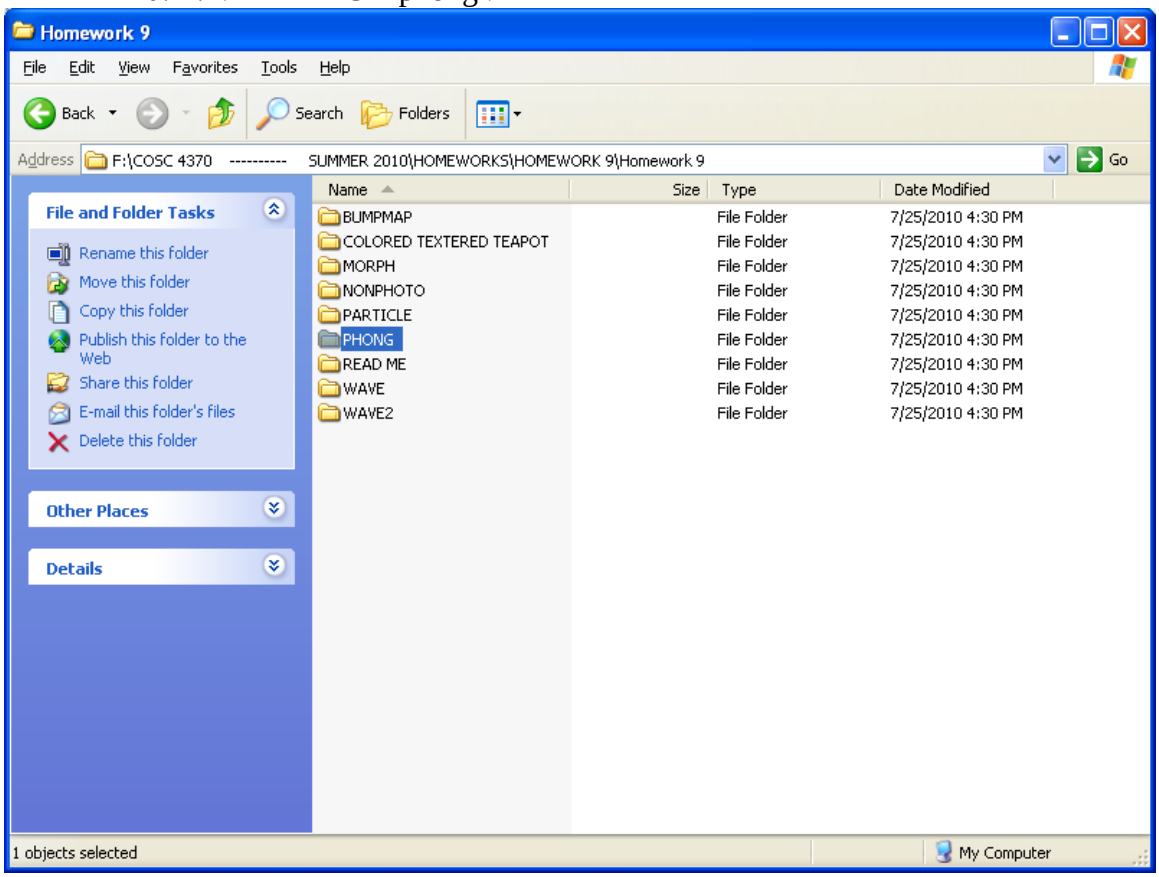
```
static void initShader(const GLchar* vShaderFile, const GLchar*
fShaderFile)
{
    GLint status;
    GLchar *vSource, *fSource;
    GLuint vShader, fShader;

    /* read shader files */

    vSource = readShaderSource(vShaderFile);
    if(vSource==NULL)
    {
        printf( "Failed to read vertex shader \n");
        exit(EXIT_FAILURE);
    }

    fSource = readShaderSource(fShaderFile);
    if(fSource==NULL)
    {
        printf("Failed to read fragment shader \n");
        exit(EXIT_FAILURE);
    }

    /* create program and shader objects */

    vShader = glCreateShader(GL_VERTEX_SHADER);
    fShader = glCreateShader(GL_FRAGMENT_SHADER);
    program = glCreateProgram();

    /* attach shaders to the program object */

    glAttachShader(program, vShader);
    glAttachShader(program, fShader);

    /* read shaders */

    glShaderSource(vShader, 1, (const GLchar**) &vSource, NULL);
    glShaderSource(fShader, 1, (const GLchar**) &fSource, NULL);

    /* compile shaders */

    glCompileShader(vShader);
    glCompileShader(fShader);

    /* error check */

    glGetShaderiv(vShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the vertex shader.");

    glGetShaderiv(fShader, GL_COMPILE_STATUS, &status);
    if(status==GL_FALSE)
    {
       printf("Failed to compile the fragment shader.\n");
       glGetShaderiv(fShader, GL_INFO_LOG_LENGTH, &elength);
       ebuffer = malloc(elength*sizeof(char));
       glGetShaderInfoLog(fShader, elength, NULL, ebuffer);
       printf("%s\n", ebuffer);
```

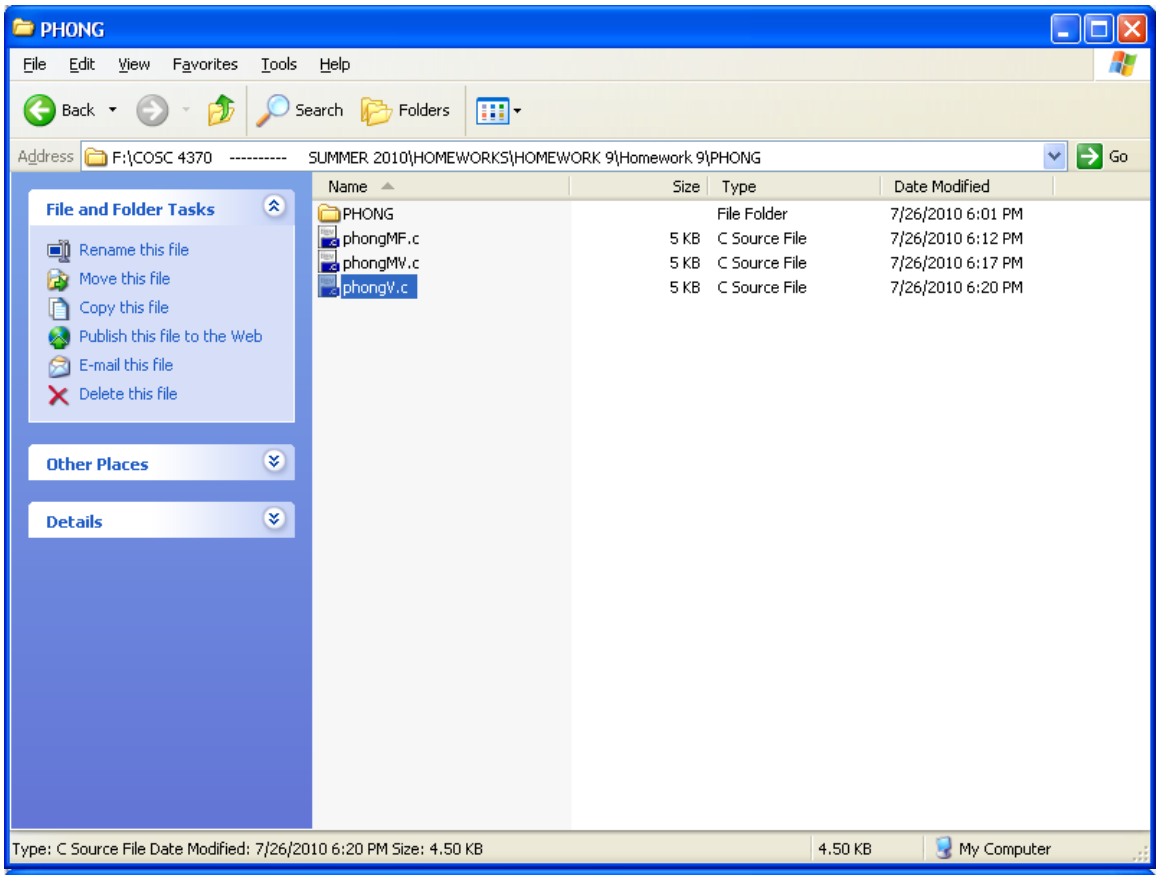
```
        exit(EXIT_FAILURE);
    }

    /* link */

    glLinkProgram(program);
    glGetProgramiv(program, GL_LINK_STATUS, &status);
    checkError(status, "Failed to link the shader program object.");

    /* use program object */

    glUseProgram(program);

    /* set up uniform parameter */

    timeParam = glGetUniformLocation(program, "time");
    vxParam = glGetAttribLocation(program, "vx");
    vyParam = glGetAttribLocation(program, "vy");
}

static void draw(void)
{
    int i;
    /* send elapsed time to shaders */

    glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glBegin(GL_POINTS);
       for(i=0; i<N; i++)
       {
           glVertexAttrib1f(vxParam, velocity[i][0]);
           glVertexAttrib1f(vyParam, velocity[i][1]);
           glVertex2f(0.5, 0.5);
       }
    glEnd();

    glutSwapBuffers();
}

static void reshape(int w, int h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,1.0,0.0,1.0);

    glViewport(0, 0, w, h);
}

static void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 27:
    case 'Q':
    case 'q':
        exit(EXIT_SUCCESS);
        break;
```

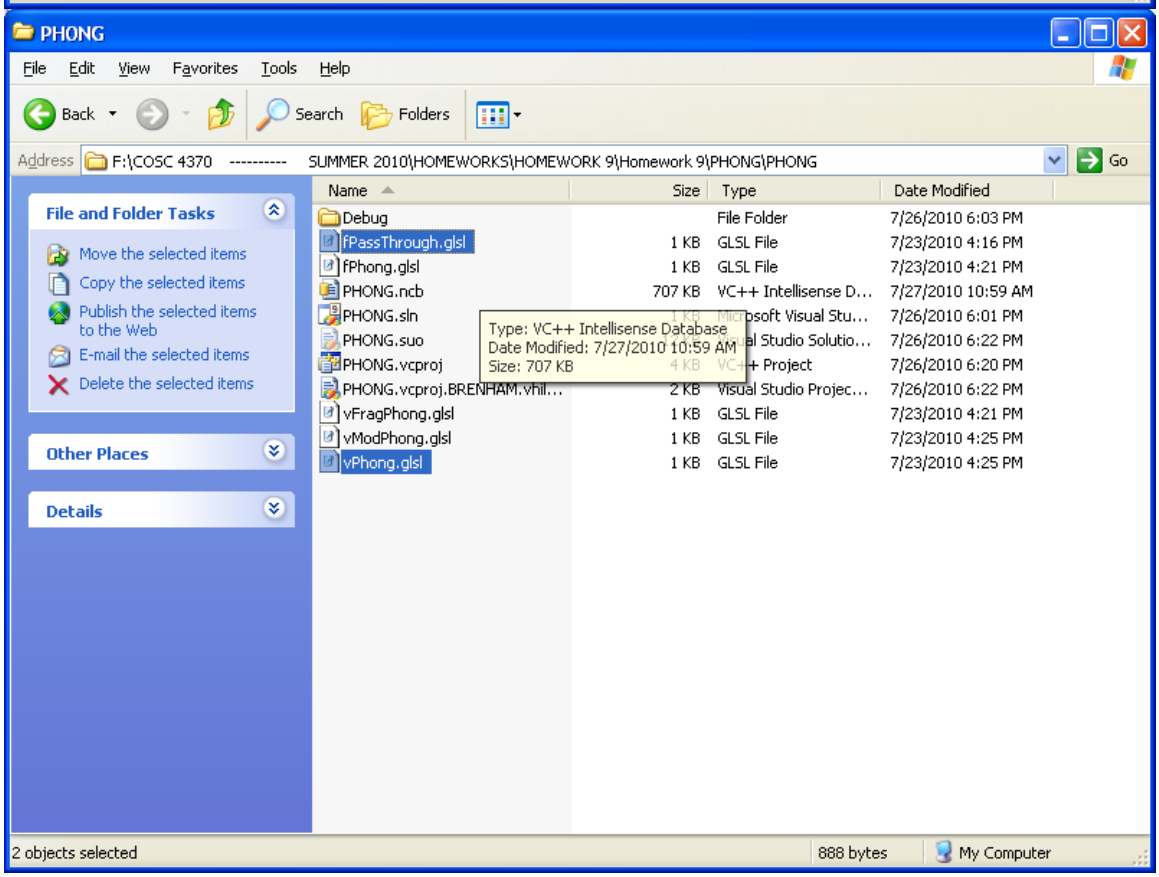
```
    default:
        break;
    }
}

void idle()
{
   glutPostRedisplay();
}

int main(int argc, char** argv)
{

    int i;
    for(i=0;i<N;i++)
    {
      velocity[i][0]=2.0*((rand()%256)/256.0-0.5);
      velocity[i][1]=2.0*((rand()%256)/256.0-0.5);
    }

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(512, 512);
    glutCreateWindow("PARTICLE GLSL example");
    glutDisplayFunc(draw);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutIdleFunc(idle);

    init();
      glewInit();
    initShader("vparticle.glsl", "fPassThrough.glsl");

    glutMainLoop();
    return 0;
}

// vparticle.glsl

uniform float time;
attribute  float vx, vy;

void main()
{
    float a = -0.00000001;
    vec4 t = gl_Vertex;
    t.y = gl_Vertex.y + 0.0001*vy*time + 0.5*a*time*time;
    t.x = gl_Vertex.x + 0.0001*vx*time;
    gl_Position = gl_ModelViewProjectionMatrix*t;
    gl_FrontColor =  gl_Color;
}


// fPassThrough.glsl

// Pass through fragment shader.
```

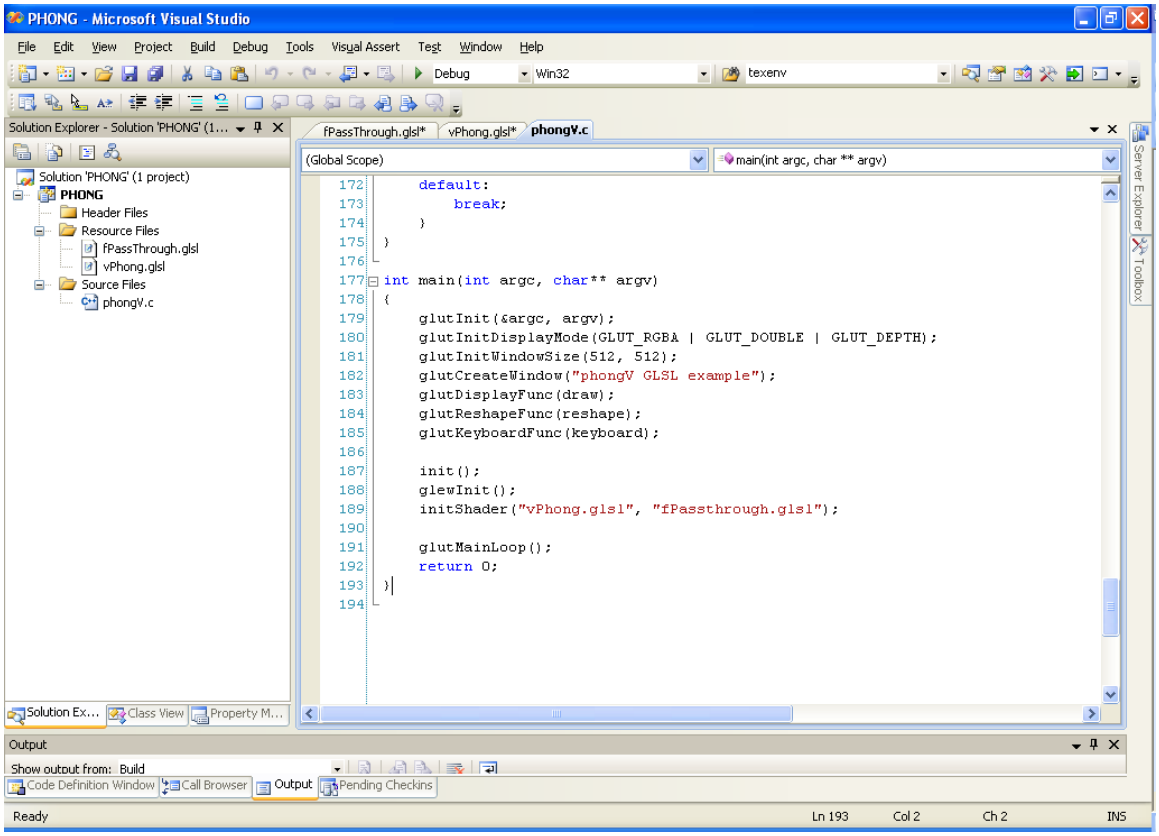
```
void main()
{
     gl_FragColor = gl_Color;
}
```

6. PHONG
    a. : VARIATION phongV




PHONG
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\PHONG
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places
Details
Name Size Type Date Modified
PHONG File Folder 7/26/2010 6:01 PM
phongMF.c 5 KB C Source File 7/26/2010 6:12 PM
phongMV.c 5 KB C Source File 7/26/2010 6:17 PM
phongV.c 5 KB C Source File 7/26/2010 6:20 PM
Type: C Source File Date Modified: 7/26/2010 6:20 PM Size: 4.50 KB
4.50 KB
My Computer


PHONG
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\PHONG\PHONG
Go
File and Folder Tasks
Move the selected items
Copy the selected items
Publish the selected items to the Web
E-mail the selected items
Delete the selected items
Other Places
Details
Name Size Type Date Modified
Debug File Folder 7/26/2010 6:03 PM
fPassThrough.glsl 1 KB GLSL File 7/23/2010 4:16 PM
fPhong.glsl 1 KB GLSL File 7/23/2010 4:21 PM
PHONG.ncb 707 KB VC++ Intellisense D... 7/27/2010 10:59 AM
PHONG.sln Microsoft Visual Stu... 7/26/2010 6:01 PM
PHONG.suo Studio Solutio... 7/26/2010 6:22 PM
PHONG.vcproj Project 7/26/2010 6:20 PM
PHONG.vcproj.BRENHAM.vhil... Visual Studio Projec... 7/26/2010 6:22 PM
vFragPhong.glsl 1 KB GLSL File 7/23/2010 4:21 PM
vModPhong.glsl 1 KB GLSL File 7/23/2010 4:25 PM
vPhong.glsl 1 KB GLSL File 7/23/2010 4:25 PM
Type: VC++ Intellisense Database
Date Modified: 7/27/2010 10:59 AM
Size: 707 KB
2 objects selected
888 bytes
My Computer


PHONG - Microsoft Visual Studio
File Edit View Project Build Debug Tools Visual Assert Test Window Help
Debug
Win32
texenv
Solution Explorer - Solution 'PHONG' (1...
Solution 'PHONG' (1 project)
PHONG
Header Files
Resource Files
fPassThrough.glsl
vPhong.glsl
Source Files
phongV.c
fPassThrough.glsl*
vPhong.glsl*
phongV.c
(Global Scope)
main(int argc, char ** argv)
default:
break;
}
}
int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
glutInitWindowSize(512, 512);
glutCreateWindow("phongV GLSL example");
glutDisplayFunc(draw);
glutReshapeFunc(reshape);
glutKeyboardFunc(keyboard);
init();
glewInit();
initShader("vPhong.glsl", "fPassthrough.glsl");
glutMainLoop();
return 0;
}
Server Explorer
Toolbox
Solution Ex...
Class View
Property M...
Output
Show output from: Build
Code Definition Window
Call Browser
Output
Pending Checkins
Ready
Ln 193
Col 2
Ch 2
INS


```
//phongV.c


/* display teapot with vertex and fragment shaders */
/* sets up elapsed time parameter for use by shaders */

#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <GL/glew.h>
#include <GL/glut.h>


const GLdouble nearVal     = 1.0;
const GLdouble farVal      = 20.0;
const GLfloat  lightPos[4] = {3.0f, 3.0f, 3.0f, 1.0f};
GLuint         program     = 0;
GLint          timeParam;
```

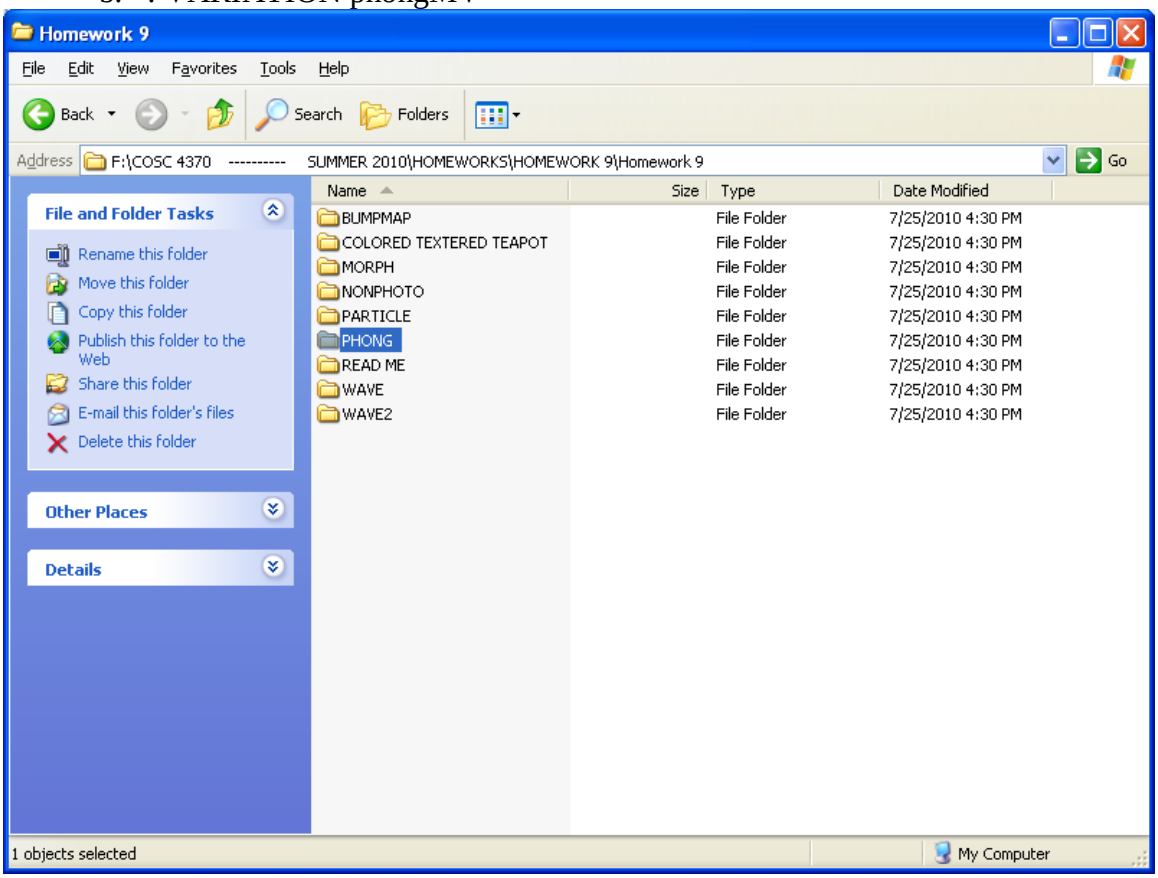
```
/* shader reader */
/* creates null terminated string from file */

static char* readShaderSource(const char* shaderFile)
{
    FILE* fp = fopen(shaderFile, "rb");

    char* buf;
    long size;

    if(fp==NULL) return NULL;
    fseek(fp, 0L, SEEK_END);
    size = ftell(fp);
    fseek(fp, 0L, SEEK_SET);
    buf = (char*) malloc((size+1) * sizeof(char));
    fread(buf, 1, size, fp);
      buf[size] = '\0';
    fclose(fp);
    return buf;
}

/* error printing function */

static void checkError(GLint status, const char *msg)
{
    if (!status)
    {
        printf("%s\n", msg);
        exit(EXIT_FAILURE);
    }
}

/* standard OpenGL initialization */

static void init()
{
    const float teapotColor[]     = {0.3f, 0.5f, 0.4f, 1.0f};
    const float teapotSpecular[]  = {0.8f, 0.8f, 0.8f, 1.0f};
    const float teapotShininess[] = {80.0f};

    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, teapotColor);
    glMaterialfv(GL_FRONT, GL_SPECULAR, teapotSpecular);
    glMaterialfv(GL_FRONT, GL_SHININESS, teapotShininess);

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double) glutGet(GLUT_WINDOW_WIDTH) / (double)
glutGet(GLUT_WINDOW_HEIGHT), nearVal, farVal);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
```

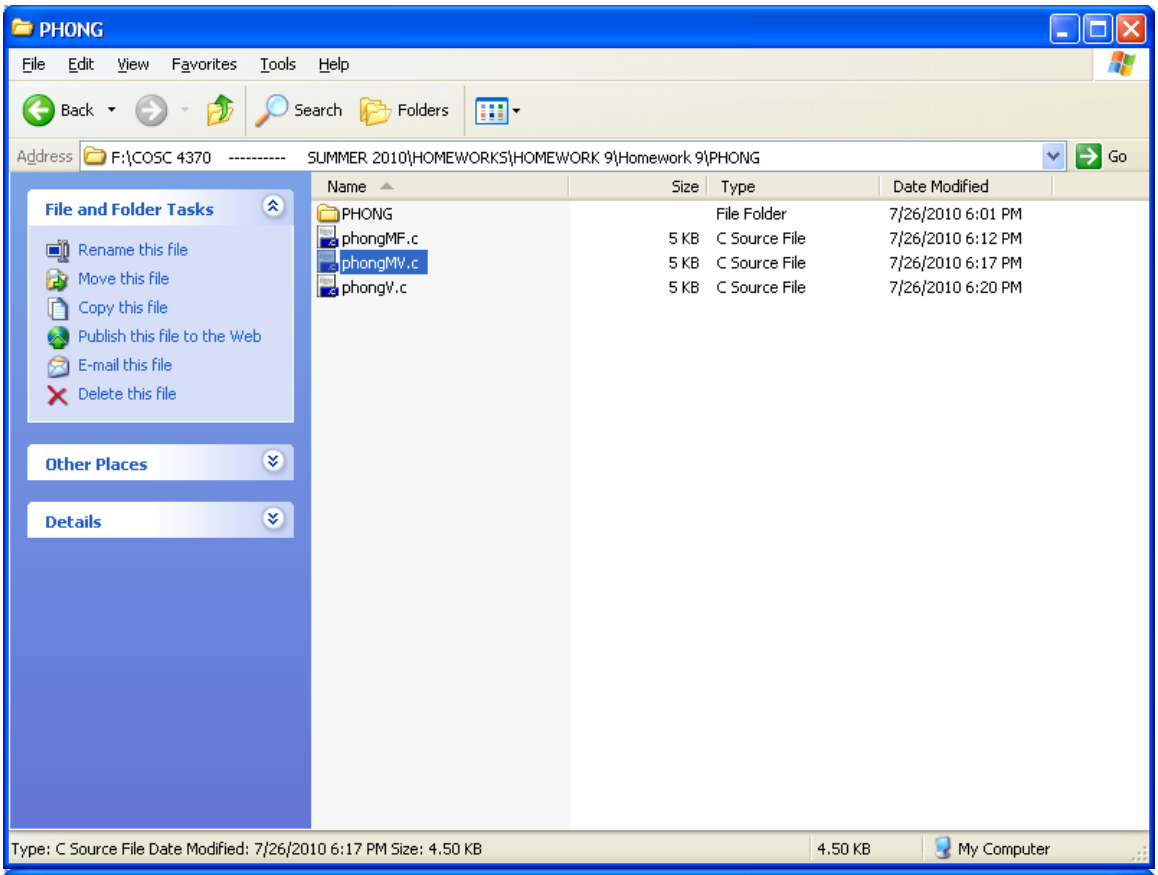
```
}

/* GLSL initialization */

static void initShader(const GLchar* vShaderFile, const GLchar*
fShaderFile)
{
    GLint status = glGetError()==GL_NO_ERROR;
    GLchar *vSource, *fSource;
    GLuint vShader, fShader;

    /* read shader files */

    vSource = readShaderSource(vShaderFile);
    checkError(status, "Failed to read vertex shader");

    fSource = readShaderSource(fShaderFile);
    checkError(status, "Failed to read fragment shader");

    /* create program and shader objects */

    vShader = glCreateShader(GL_VERTEX_SHADER);
    fShader = glCreateShader(GL_FRAGMENT_SHADER);
    program = glCreateProgram();

    /* attach shaders to the program object */

    glAttachShader(program, vShader);
    glAttachShader(program, fShader);

    /* read shaders */

    glShaderSource(vShader, 1, (const GLchar**) &vSource, NULL);
    glShaderSource(fShader, 1, (const GLchar**) &fSource, NULL);

    /* compile shaders */

    glCompileShader(vShader);
    glCompileShader(fShader);

    /* error check */

    glGetShaderiv(vShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the vertex shader.");

    glGetShaderiv(fShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the fragment shader.");

    /* link */

    glLinkProgram(program);
    glGetProgramiv(program, GL_LINK_STATUS, &status);
    checkError(status, "Failed to link the shader program object.");

    /* use program object */

    glUseProgram(program);
```

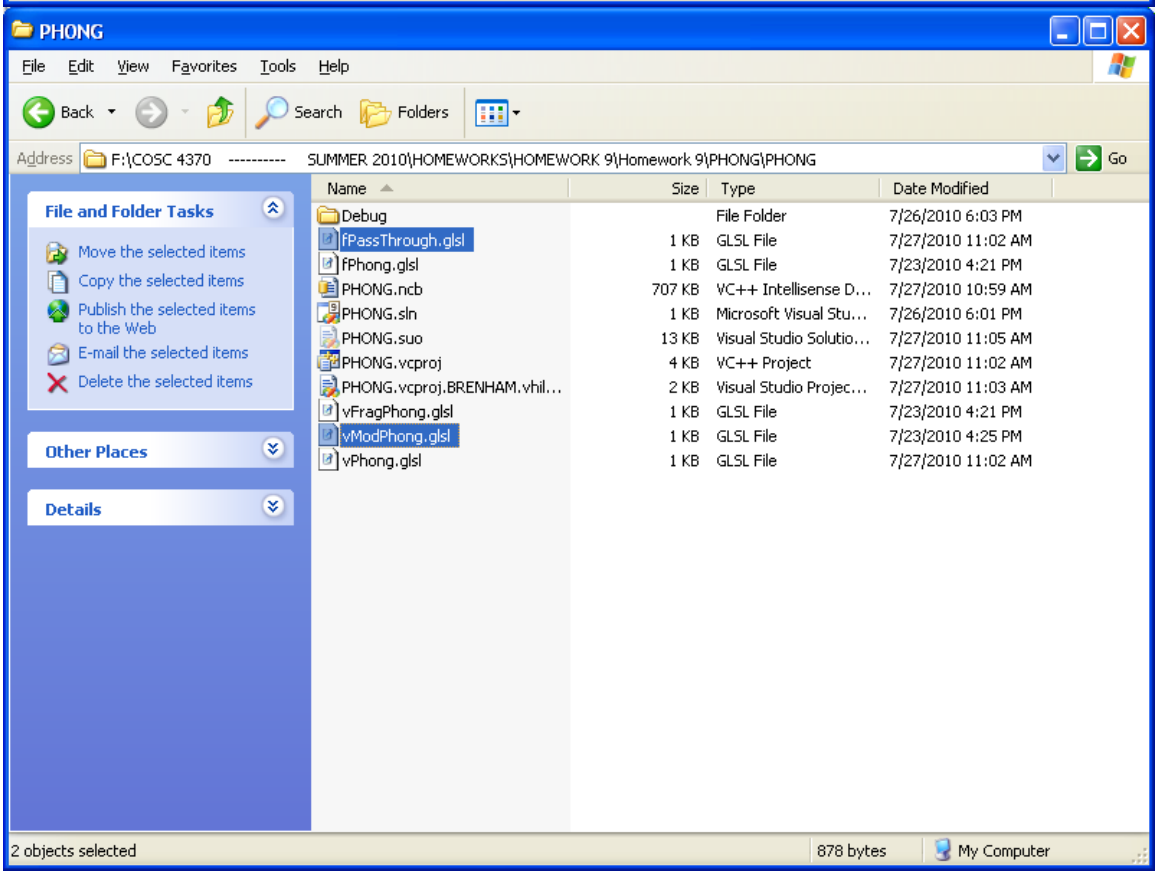
```
    /* set up uniform parameter */

    timeParam = glGetUniformLocation(program, "time");
}

static void draw(void)
{
    /* send elapsed time to shaders */

    glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glTranslatef(0.0f, 0.0f, -5.0f);
    glutSolidTeapot(1.0);
    glPopMatrix();
    glutSwapBuffers();
}

static void reshape(int w, int h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double) w / (double) h, nearVal, farVal);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glViewport(0, 0, w, h);

    glutPostRedisplay();
}

static void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 27:
    case 'Q':
    case 'q':
        exit(EXIT_SUCCESS);
        break;
    default:
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(512, 512);
    glutCreateWindow("phongV GLSL example");
    glutDisplayFunc(draw);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);

    init();
      glewInit();
```

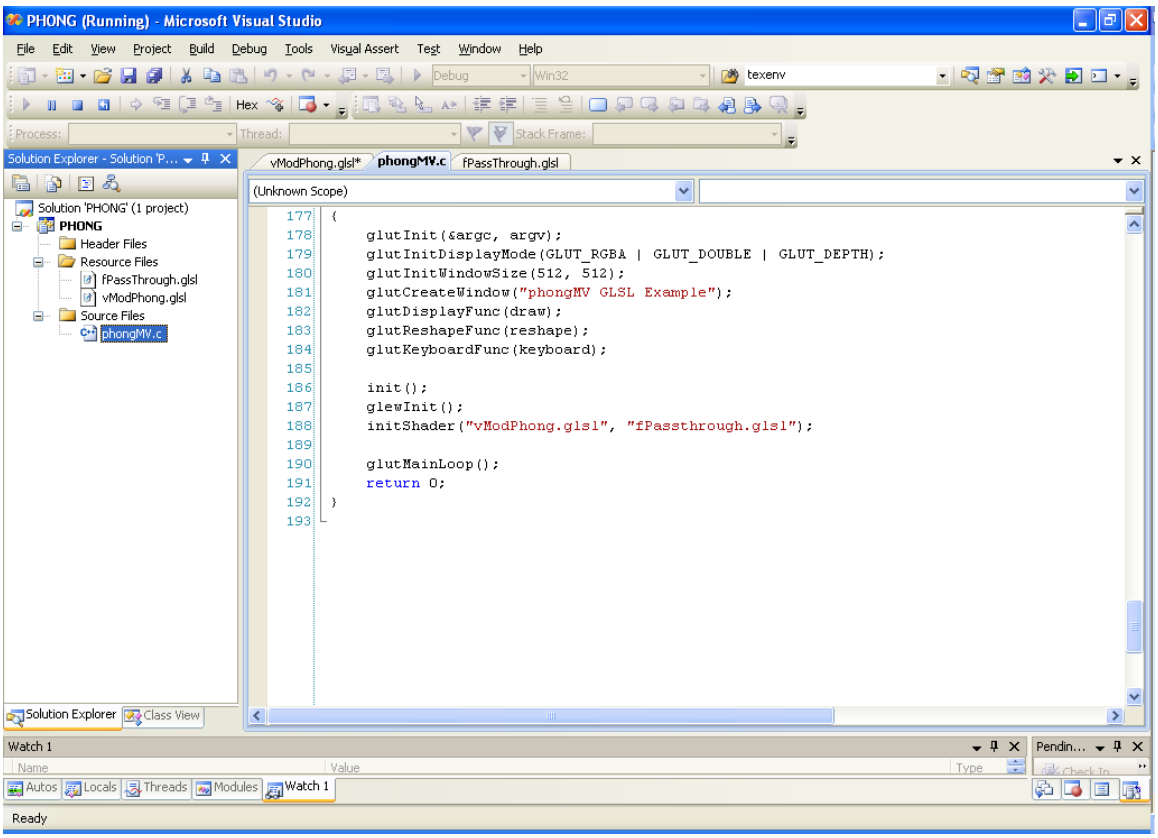
```
    initShader("vPhong.glsl", "fPassthrough.glsl");

    glutMainLoop();
    return 0;
}
// vPhong.glsl

void main()
{
    float factor = 1.0;
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;

    vec4 eyePosition = gl_ModelViewMatrix * gl_Vertex;
    vec4 eyeLightPos = gl_LightSource[0].position;
    vec3 eyeNormalVec = normalize(gl_NormalMatrix * gl_Normal);
    vec3 eyeLightVec = normalize(eyeLightPos.xyz - eyePosition.xyz);
    vec3 eyeViewVec = -normalize(eyePosition.xyz);
    vec3 eyeReflectVec = -reflect(eyeLightVec, eyeNormalVec);
    float Kd = max(dot(eyeLightVec, eyeNormalVec), 0.0);
    float Ks = pow(max(dot(eyeViewVec, eyeReflectVec),0.0),
factor*gl_FrontMaterial.shininess);
    float Ka = 1.0;

    gl_FrontColor = Kd * gl_FrontLightProduct[0].diffuse + Ks *
gl_FrontLightProduct[0].specular +
gl_FrontLightModelProduct.sceneColor;
}


// fPassThrough.glsl

// Pass through fragment shader.

void main()
{
    gl_FragColor = gl_Color;
}
```

b. : VARIATION phongMV




PHONG
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\PHONG
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places
Details
Name Size Type Date Modified
PHONG File Folder 7/26/2010 6:01 PM
phongMF.c 5 KB C Source File 7/26/2010 6:12 PM
phongMV.c 5 KB C Source File 7/26/2010 6:17 PM
phongV.c 5 KB C Source File 7/26/2010 6:20 PM
Type: C Source File Date Modified: 7/26/2010 6:17 PM Size: 4.50 KB
4.50 KB
My Computer


PHONG
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\PHONG\PHONG
Go
File and Folder Tasks
Move the selected items
Copy the selected items
Publish the selected items to the Web
E-mail the selected items
Delete the selected items
Other Places
Details
Name Size Type Date Modified
Debug File Folder 7/26/2010 6:03 PM
fPassThrough.glsl 1 KB GLSL File 7/27/2010 11:02 AM
fPhong.glsl 1 KB GLSL File 7/23/2010 4:21 PM
PHONG.ncb 707 KB VC++ Intellisense D... 7/27/2010 10:59 AM
PHONG.sln 1 KB Microsoft Visual Stu... 7/26/2010 6:01 PM
PHONG.suo 13 KB Visual Studio Solutio... 7/27/2010 11:05 AM
PHONG.vcproj 4 KB VC++ Project 7/27/2010 11:02 AM
PHONG.vcproj.BRENHAM.vhil... 2 KB Visual Studio Projec... 7/27/2010 11:03 AM
vFragPhong.glsl 1 KB GLSL File 7/23/2010 4:21 PM
vModPhong.glsl 1 KB GLSL File 7/23/2010 4:25 PM
vPhong.glsl 1 KB GLSL File 7/27/2010 11:02 AM
2 objects selected
878 bytes
My Computer


PHONG (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Tools Visual Assert Test Window Help
Debug
Win32
texenv
Hex
Process:
Thread:
Stack Frame:
Solution Explorer - Solution 'P...
Solution 'PHONG' (1 project)
PHONG
Header Files
Resource Files
fPassThrough.glsl
vModPhong.glsl
Source Files
phongMV.c
Solution Explorer
Class View
vModPhong.glsl*
phongMV.c
fPassThrough.glsl
(Unknown Scope)
177 {
178 glutInit(&argc, argv);
179 glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
180 glutInitWindowSize(512, 512);
181 glutCreateWindow("phongMV GLSL Example");
182 glutDisplayFunc(draw);
183 glutReshapeFunc(reshape);
184 glutKeyboardFunc(keyboard);
185
186 init();
187 glewInit();
188 initShader("vModPhong.glsl", "fPassthrough.glsl");
189
190 glutMainLoop();
191 return 0;
192 }
193
Watch 1
Name
Value
Type
Autos
Locals
Threads
Modules
Watch 1
Ready


```
// phoneMV.c

/* display teapot with vertex and fragment shaders */
/* sets up elapsed time parameter for use by shaders */

#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <GL/glew.h>
#include <GL/glut.h>


const GLdouble nearVal     = 1.0;
const GLdouble farVal      = 20.0;
const GLfloat  lightPos[4] = {3.0f, 3.0f, 3.0f, 1.0f};
GLuint         program     = 0;
GLint          timeParam;

/* shader reader */
```

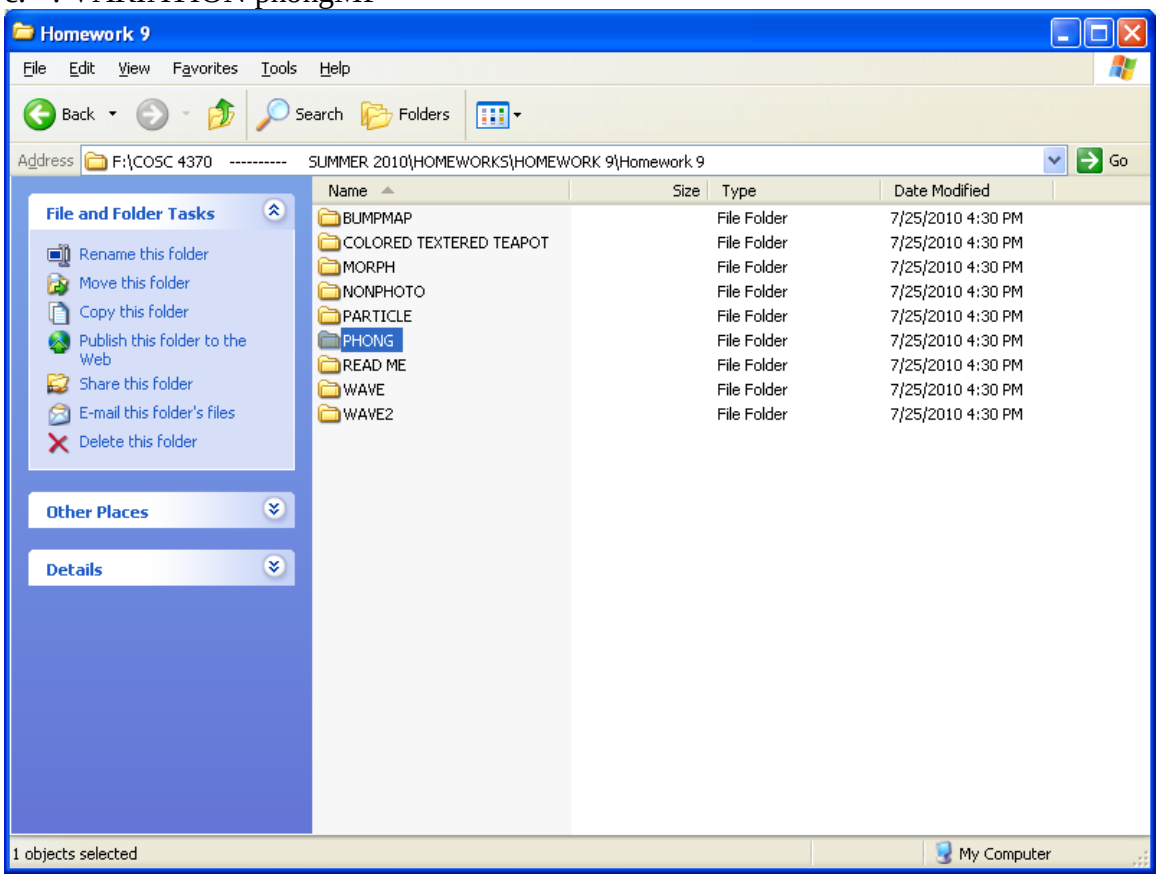
```
/* creates null terminated string from file */

static char* readShaderSource(const char* shaderFile)
{
    FILE* fp = fopen(shaderFile, "rb");

    char* buf;
    long size;

    if(fp==NULL) return NULL;
    fseek(fp, 0L, SEEK_END);
    size = ftell(fp);
    fseek(fp, 0L, SEEK_SET);
    buf = (char*) malloc((size+1) * sizeof(char));
    fread(buf, 1, size, fp);
      buf[size] = '\0';
    fclose(fp);
    return buf;
}

/* error printing function */

static void checkError(GLint status, const char *msg)
{
    if (!status)
    {
        printf("%s\n", msg);
        exit(EXIT_FAILURE);
    }
}

/* standard OpenGL initialization */

static void init()
{
    const float teapotColor[]     = {0.3f, 0.5f, 0.4f, 1.0f};
    const float teapotSpecular[]  = {0.8f, 0.8f, 0.8f, 1.0f};
    const float teapotShininess[] = {80.0f};

    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, teapotColor);
    glMaterialfv(GL_FRONT, GL_SPECULAR, teapotSpecular);
    glMaterialfv(GL_FRONT, GL_SHININESS, teapotShininess);

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double) glutGet(GLUT_WINDOW_WIDTH) / (double)
glutGet(GLUT_WINDOW_HEIGHT), nearVal, farVal);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
}
```

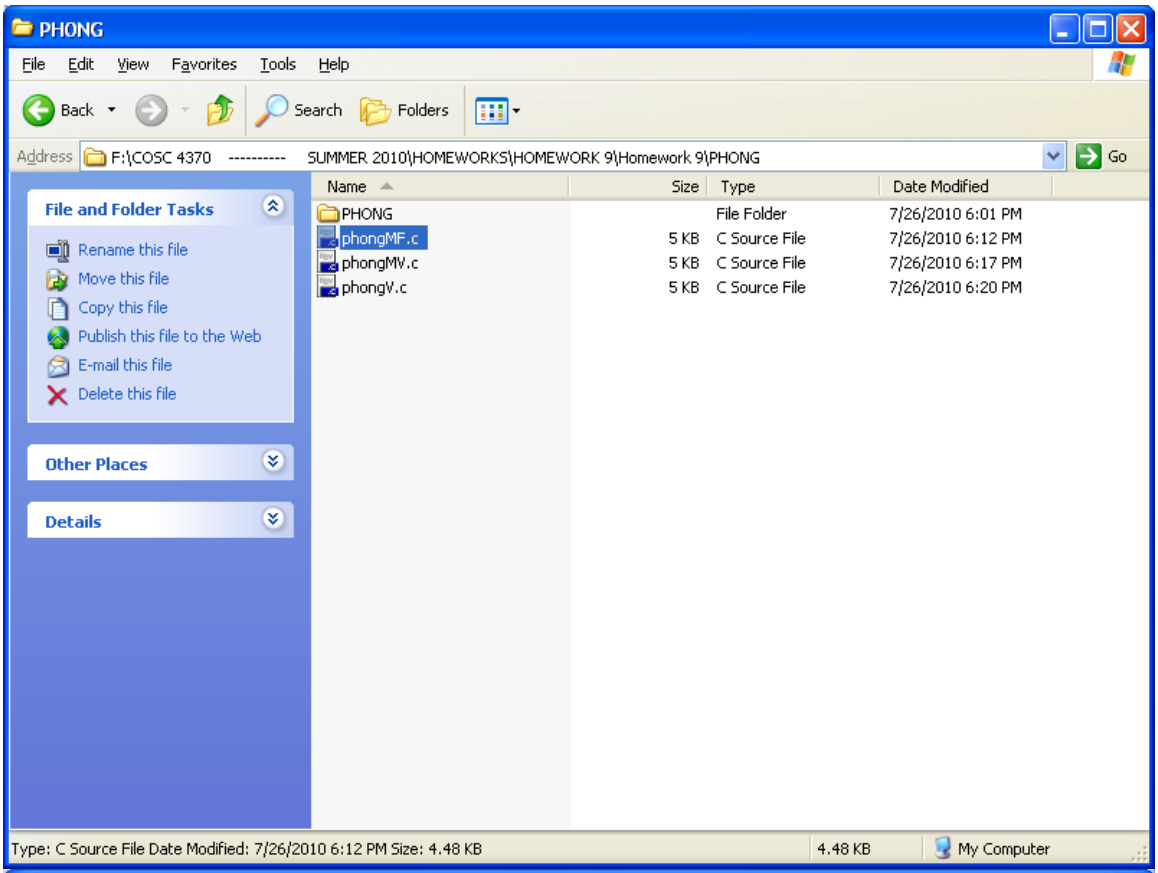
```
/* GLSL initialization */

static void initShader(const GLchar* vShaderFile, const GLchar*
fShaderFile)
{
    GLint status = glGetError()==GL_NO_ERROR;
    GLchar *vSource, *fSource;
    GLuint vShader, fShader;

    /* read shader files */

    vSource = readShaderSource(vShaderFile);
    checkError(status, "Failed to read vertex shader");

    fSource = readShaderSource(fShaderFile);
    checkError(status, "Failed to read fragment shader");

    /* create program and shader objects */

    vShader = glCreateShader(GL_VERTEX_SHADER);
    fShader = glCreateShader(GL_FRAGMENT_SHADER);
    program = glCreateProgram();

    /* attach shaders to the program object */

    glAttachShader(program, vShader);
    glAttachShader(program, fShader);

    /* read shaders */

    glShaderSource(vShader, 1, (const GLchar**) &vSource, NULL);
    glShaderSource(fShader, 1, (const GLchar**) &fSource, NULL);

    /* compile shaders */

    glCompileShader(vShader);
    glCompileShader(fShader);

    /* error check */

    glGetShaderiv(vShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the vertex shader.");

    glGetShaderiv(fShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the fragment shader.");

    /* link */

    glLinkProgram(program);
    glGetProgramiv(program, GL_LINK_STATUS, &status);
    checkError(status, "Failed to link the shader program object.");

    /* use program object */

    glUseProgram(program);
```

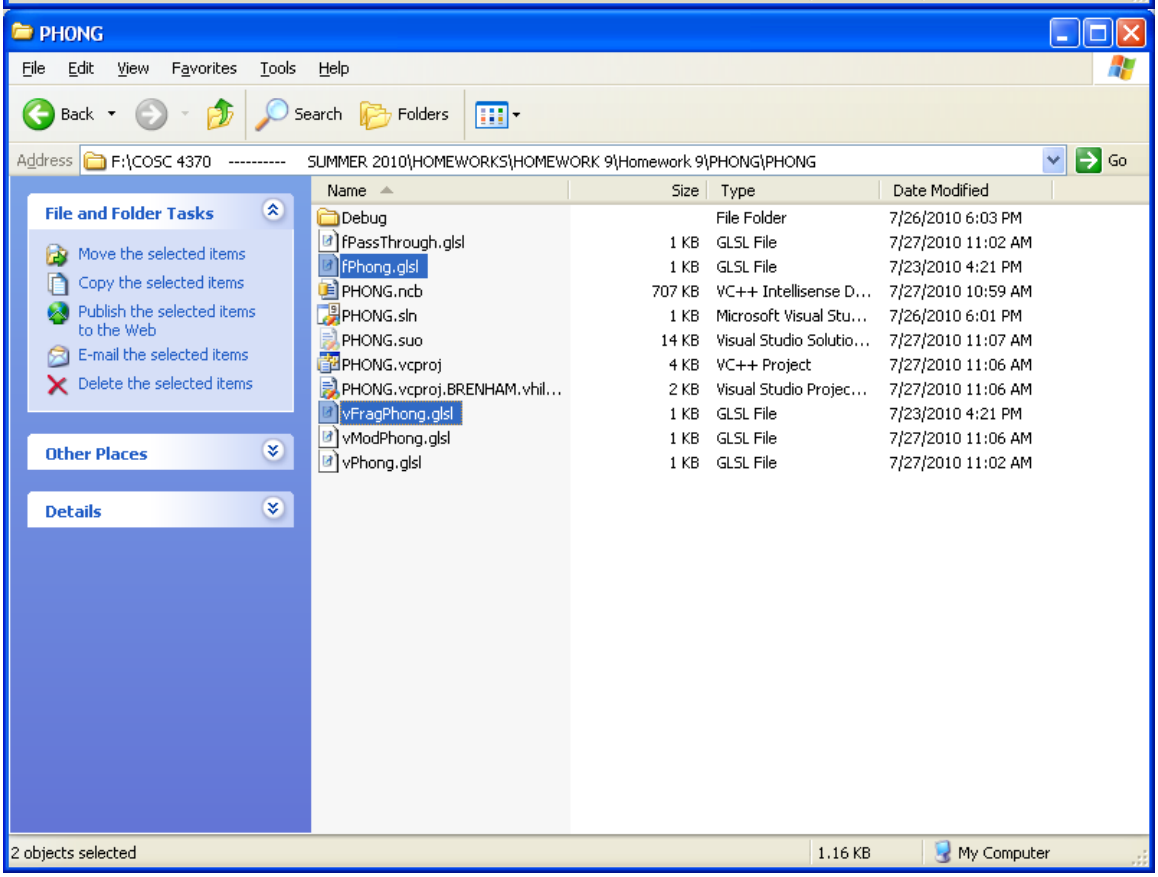
```
    /* set up uniform parameter */

    timeParam = glGetUniformLocation(program, "time");
}

static void draw(void)
{
    /* send elapsed time to shaders */

    glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glTranslatef(0.0f, 0.0f, -5.0f);
    glutSolidTeapot(1.0);
    glPopMatrix();
    glutSwapBuffers();
}

static void reshape(int w, int h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double) w / (double) h, nearVal, farVal);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glViewport(0, 0, w, h);

    glutPostRedisplay();
}

static void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 27:
    case 'Q':
    case 'q':
        exit(EXIT_SUCCESS);
        break;
    default:
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(512, 512);
    glutCreateWindow("phongMV GLSL Example");
    glutDisplayFunc(draw);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);

    init();
      glewInit();
    initShader("vModPhong.glsl", "fPassthrough.glsl");
```

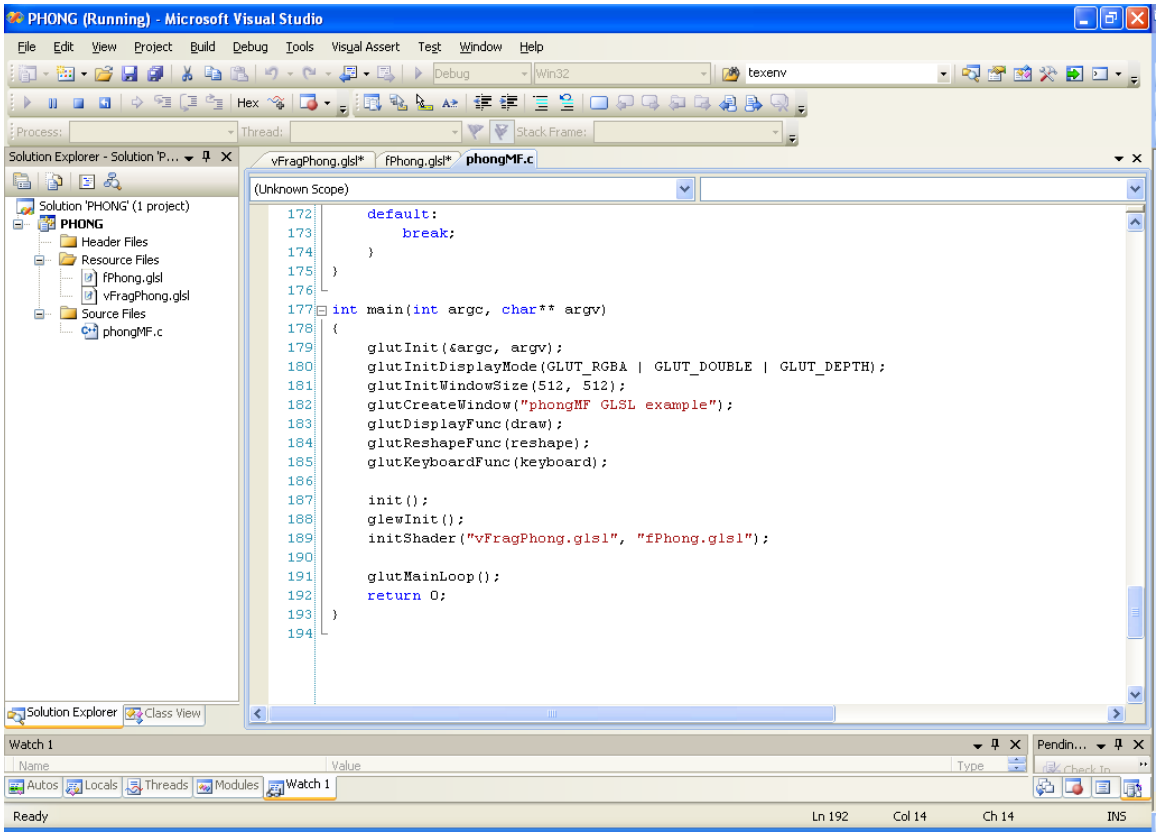
```
    glutMainLoop();
    return 0;
}

// vModPhong.glsl

void main()
{
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;

    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    vec4 eyePosition = gl_ModelViewMatrix * gl_Vertex;
    vec4 eyeLightPos = gl_LightSource[0].position;
    vec3 N = normalize(gl_NormalMatrix * gl_Normal);
    vec3 L = normalize(eyeLightPos.xyz - eyePosition.xyz);
    vec3 E = -normalize(eyePosition.xyz);
    vec3 H = normalize(L + E);
    float Kd = max(dot(L, N), 0.0);
    float Ks = pow(max(dot(N, H), 0.0), gl_FrontMaterial.shininess);
    float Ka = 0.0;

    ambient = Ka*gl_FrontLightProduct[0].ambient;
    diffuse = Kd*gl_FrontLightProduct[0].diffuse;
    specular = Ks*gl_FrontLightProduct[0].specular;
    gl_FrontColor = ambient+diffuse+specular;
}


// fPassThrough.glsl

// Pass through fragment shader.

void main()
{
    gl_FragColor = gl_Color;
}
```

c. : VARIATION phongMF




PHONG
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\PHONG
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places
Details
Name Size Type Date Modified
PHONG File Folder 7/26/2010 6:01 PM
phongMF.c 5 KB C Source File 7/26/2010 6:12 PM
phongMV.c 5 KB C Source File 7/26/2010 6:17 PM
phongV.c 5 KB C Source File 7/26/2010 6:20 PM
Type: C Source File Date Modified: 7/26/2010 6:12 PM Size: 4.48 KB
4.48 KB
My Computer


PHONG
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\PHONG\PHONG
Go
File and Folder Tasks
Move the selected items
Copy the selected items
Publish the selected items to the Web
E-mail the selected items
Delete the selected items
Other Places
Details
Name Size Type Date Modified
Debug File Folder 7/26/2010 6:03 PM
fPassThrough.glsl 1 KB GLSL File 7/27/2010 11:02 AM
fPhong.glsl 1 KB GLSL File 7/23/2010 4:21 PM
PHONG.ncb 707 KB VC++ Intellisense D... 7/27/2010 10:59 AM
PHONG.sln 1 KB Microsoft Visual Stu... 7/26/2010 6:01 PM
PHONG.suo 14 KB Visual Studio Solutio... 7/27/2010 11:07 AM
PHONG.vcproj 4 KB VC++ Project 7/27/2010 11:06 AM
PHONG.vcproj.BRENHAM.vhil... 2 KB Visual Studio Projec... 7/27/2010 11:06 AM
vFragPhong.glsl 1 KB GLSL File 7/23/2010 4:21 PM
vModPhong.glsl 1 KB GLSL File 7/27/2010 11:06 AM
vPhong.glsl 1 KB GLSL File 7/27/2010 11:02 AM
2 objects selected
1.16 KB
My Computer


PHONG (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Tools Visual Assert Test Window Help
Solution Explorer - Solution 'P...
Solution 'PHONG' (1 project)
PHONG
Header Files
Resource Files
fPhong.glsl
vFragPhong.glsl
Source Files
phongMF.c
vFragPhong.glsl*
fPhong.glsl*
phongMF.c
(Unknown Scope)
172 default:
173 break;
174 }
175 }
176
177 int main(int argc, char** argv)
178 {
179 glutInit(&argc, argv);
180 glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
181 glutInitWindowSize(512, 512);
182 glutCreateWindow("phongMF GLSL example");
183 glutDisplayFunc(draw);
184 glutReshapeFunc(reshape);
185 glutKeyboardFunc(keyboard);
186
187 init();
188 glewInit();
189 initShader("vFragPhong.glsl", "fPhong.glsl");
190
191 glutMainLoop();
192 return 0;
193 }
194
Watch 1
Name
Value
Type
Autos Locals Threads Modules Watch 1
Ready
Ln 192 Col 14 Ch 14 INS


```
//phongMF.c

/* display teapot with vertex and fragment shaders */
/* sets up elapsed time parameter for use by shaders */

#include <GL/glew.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <GL/glut.h>


const GLdouble nearVal     = 1.0;
const GLdouble farVal      = 20.0;
const GLfloat  lightPos[4] = {3.0f, 3.0f, 3.0f, 1.0f};

GLuint         program;
GLint          timeParam;
```

```
/* shader reader */
/* creates null terminated string from file */

static char* readShaderSource(const char* shaderFile)
{
    FILE* fp = fopen(shaderFile, "rb");

    char* buf;
    long size;

    if(fp==NULL) return NULL;
    fseek(fp, 0L, SEEK_END);
    size = ftell(fp);
    fseek(fp, 0L, SEEK_SET);
    buf = (char*) malloc((size+1) * sizeof(char));
    fread(buf, 1, size, fp);
      buf[size] = '\0';
    fclose(fp);
    return buf;
}

/* error printing function */

static void checkError(GLint status, const char *msg)
{
    if (!status)
    {
        printf("%s\n", msg);
        exit(EXIT_FAILURE);
    }
}

/* standard OpenGL initialization */

static void init()
{
    const float teapotColor[]     = {0.3f, 0.5f, 0.4f, 1.0f};
    const float teapotSpecular[]  = {0.8f, 0.8f, 0.8f, 1.0f};
    const float teapotShininess[] = {80.0f};

    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, teapotColor);
    glMaterialfv(GL_FRONT, GL_SPECULAR, teapotSpecular);
    glMaterialfv(GL_FRONT, GL_SHININESS, teapotShininess);

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double) glutGet(GLUT_WINDOW_WIDTH) / (double)
glutGet(GLUT_WINDOW_HEIGHT), nearVal, farVal);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
```

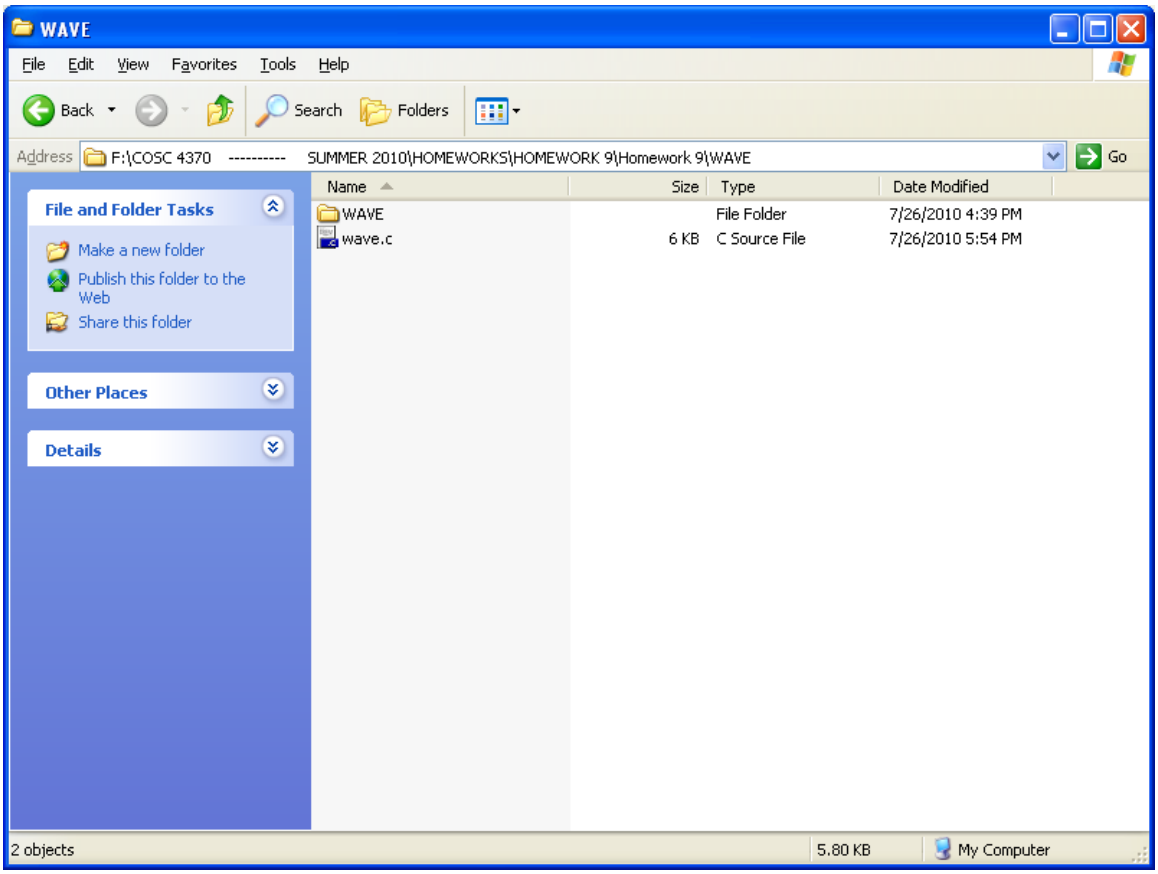
```
    glEnable(GL_DEPTH_TEST);
}

/* GLSL initialization */

static void initShader(const GLchar* vShaderFile, const GLchar*
fShaderFile)
{
    GLint status = glGetError()==GL_NO_ERROR;
    GLchar *vSource, *fSource;
    GLuint vShader, fShader;

    /* read shader files */

    vSource = readShaderSource(vShaderFile);
    checkError(status, "Failed to read vertex shader");

    fSource = readShaderSource(fShaderFile);
    checkError(status, "Failed to read fragment shader");

    /* create program and shader objects */

    vShader = glCreateShader(GL_VERTEX_SHADER);
    fShader = glCreateShader(GL_FRAGMENT_SHADER);
    program = glCreateProgram();

    /* attach shaders to the program object */

    glAttachShader(program, vShader);
    glAttachShader(program, fShader);

    /* read shaders */

    glShaderSource(vShader, 1, (const GLchar**) &vSource, NULL);
    glShaderSource(fShader, 1, (const GLchar**) &fSource, NULL);

    /* compile shaders */

    glCompileShader(vShader);
    glCompileShader(fShader);

    /* error check */

    glGetShaderiv(vShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the vertex shader.");

    glGetShaderiv(fShader, GL_COMPILE_STATUS, &status);
    checkError(status, "Failed to compile the fragment shader.");

    /* link */

    glLinkProgram(program);
      glGetProgramiv(program, GL_LINK_STATUS, &status);
    checkError(status, "Failed to link the shader program object.");

    /* use program object */
```

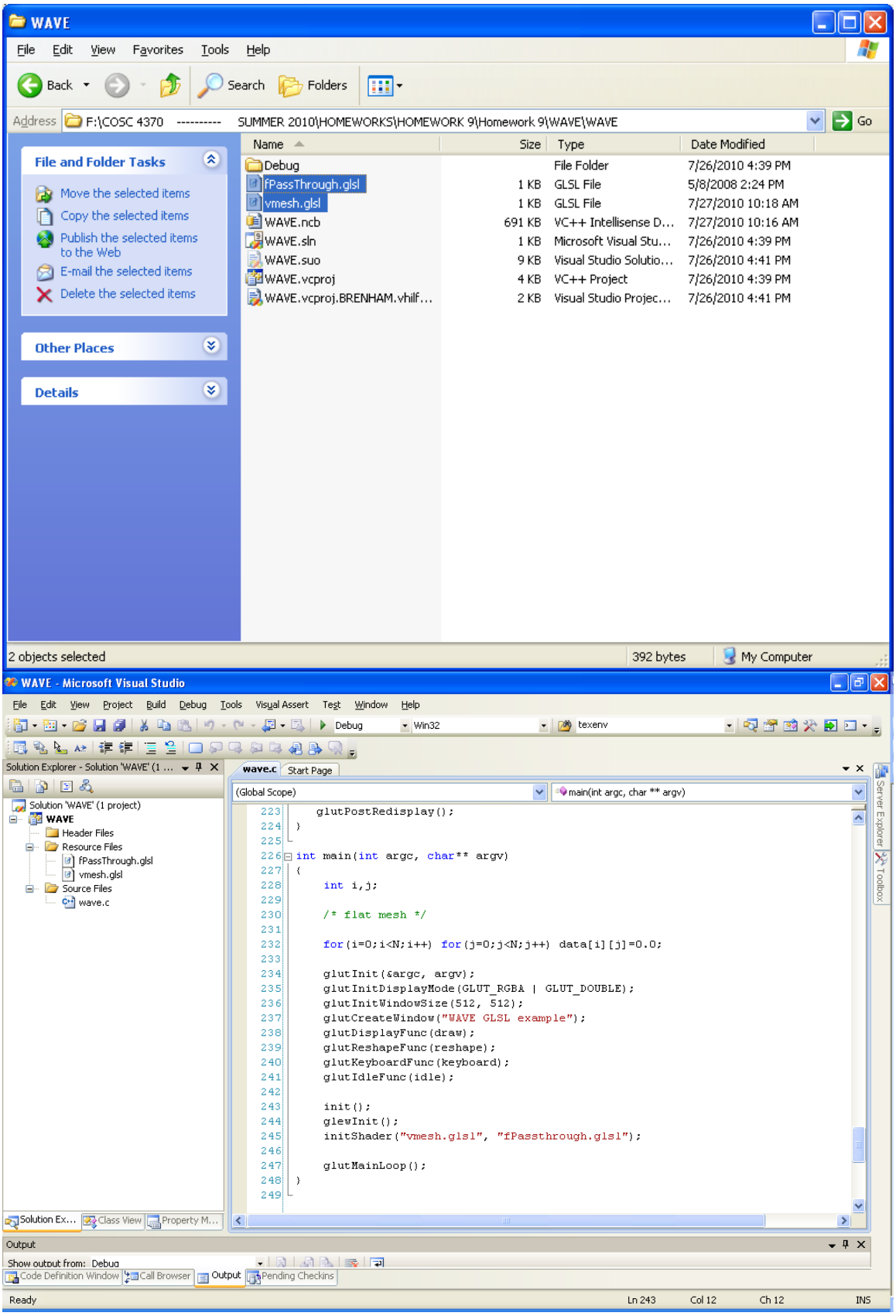
```
    glUseProgram(program);

    /* set up uniform parameter */

    timeParam = glGetUniformLocation(program, "time");
}

static void draw(void)
{
    /* send elapsed time to shaders */

    glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glTranslatef(0.0f, 0.0f, -5.0f);
    glutSolidTeapot(1.0);
    glPopMatrix();
    glutSwapBuffers();
}

static void reshape(int w, int h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double) w / (double) h, nearVal, farVal);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glViewport(0, 0, w, h);

    glutPostRedisplay();
}

static void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 27:
    case 'Q':
    case 'q':
        exit(EXIT_SUCCESS);
        break;
    default:
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(512, 512);
    glutCreateWindow("phongMF GLSL example");
    glutDisplayFunc(draw);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);

    init();
```

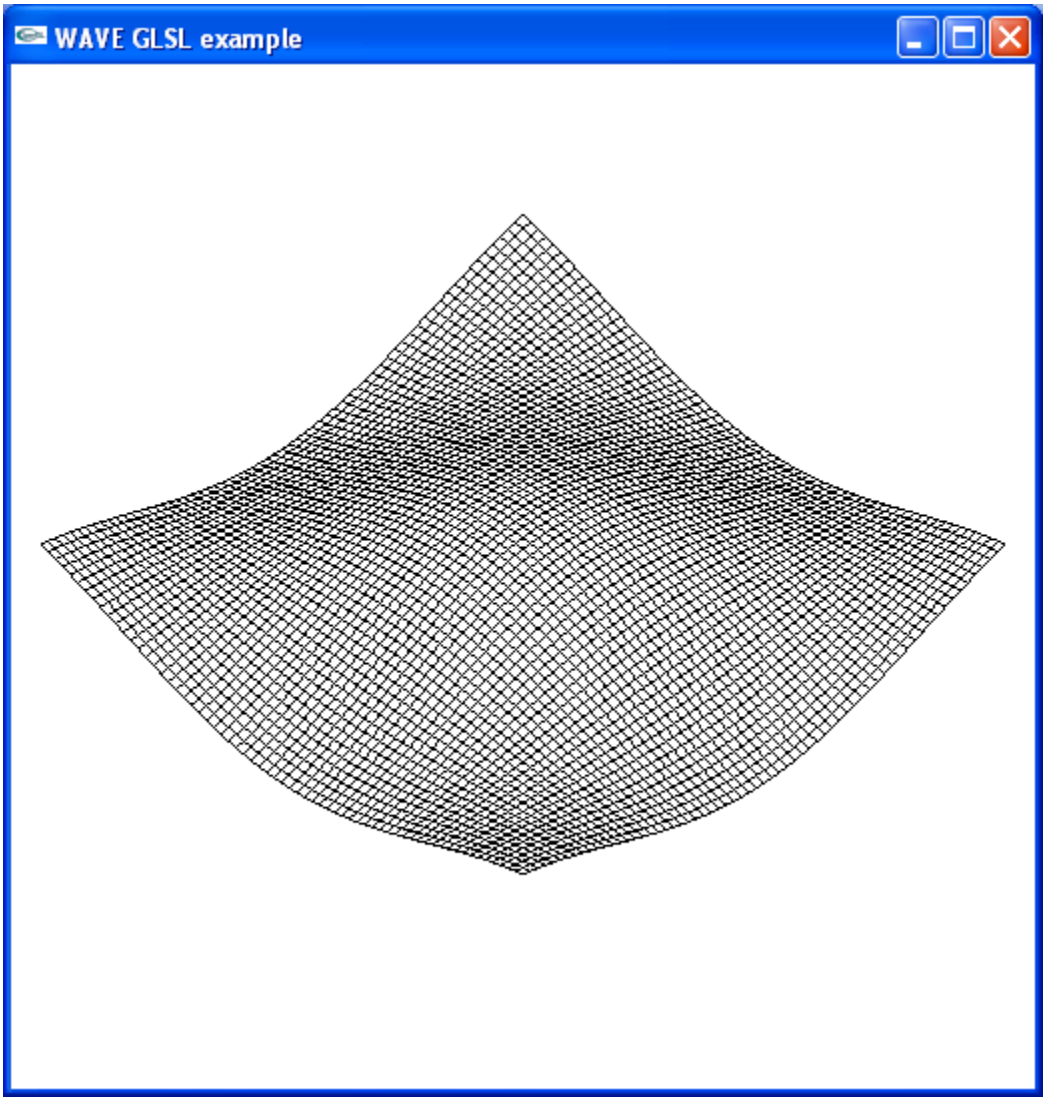
```
      glewInit();
    initShader("vFragPhong.glsl", "fPhong.glsl");

    glutMainLoop();
    return 0;
}


// vFragPhong.glsl

// Vertex shader for per-pixel Phong shading.

varying vec3 N;
varying vec3 L;
varying vec3 E;
varying vec3 H;

void main()
{
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;

    vec4 eyePosition = gl_ModelViewMatrix * gl_Vertex;
    vec4 eyeLightPos = gl_LightSource[0].position;

    N = normalize(gl_NormalMatrix * gl_Normal);
    L = normalize(eyeLightPos.xyz - eyePosition.xyz);
    E = -normalize(eyePosition.xyz);
    H = normalize(L + E);


}

// fPhong.glsl

// Fragmet shader for per-pixel Phong shading.

 varying vec3 N;
 varying vec3 L;
 varying vec3 E;
 varying vec3 H;

void main()
{
     vec3 Normal = normalize(N);
     vec3 Light  = normalize(L);
     vec3 Eye    = normalize(E);
     vec3 Half   = normalize(H);
     // vec3 Half  = normalize(L+E);


    float Kd = max(dot(Normal, Light), 0.0);
    float Ks = pow(max(dot(Half, Normal), 0.0),
gl_FrontMaterial.shininess);
    float Ka = 0.0;

    vec4 diffuse  = Kd * gl_FrontLightProduct[0].diffuse;
```

```
    vec4 specular = Ks * gl_FrontLightProduct[0].specular;
    vec4 ambient  = Ka * gl_FrontLightProduct[0].ambient;

    gl_FragColor = ambient + diffuse + specular;
}
```

7. WAVE


WAVE
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\WAVE
Go
File and Folder Tasks
Make a new folder
Publish this folder to the Web
Share this folder
Other Places
Details
Name Size Type Date Modified
WAVE File Folder 7/26/2010 4:39 PM
wave.c 6 KB C Source File 7/26/2010 5:54 PM
2 objects
5.80 KB
My Computer


WAVE
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\WAVE\WAVE
Go
File and Folder Tasks
Move the selected items
Copy the selected items
Publish the selected items to the Web
E-mail the selected items
Delete the selected items
Other Places
Details
Name Size Type Date Modified
Debug File Folder 7/26/2010 4:39 PM
fPassThrough.glsl 1 KB GLSL File 5/8/2008 2:24 PM
vmesh.glsl 1 KB GLSL File 7/27/2010 10:18 AM
WAVE.ncb 691 KB VC++ Intellisense D... 7/27/2010 10:16 AM
WAVE.sln 1 KB Microsoft Visual Stu... 7/26/2010 4:39 PM
WAVE.suo 9 KB Visual Studio Solutio... 7/26/2010 4:41 PM
WAVE.vcproj 4 KB VC++ Project 7/26/2010 4:39 PM
WAVE.vcproj.BRENHAM.vhilf... 2 KB Visual Studio Projec... 7/26/2010 4:41 PM
2 objects selected
392 bytes
My Computer
WAVE - Microsoft Visual Studio
File Edit View Project Build Debug Tools Visual Assert Test Window Help
Debug Win32 texenv
Solution Explorer - Solution 'WAVE' (1 ...
Solution 'WAVE' (1 project)
WAVE
Header Files
Resource Files
fPassThrough.glsl
vmesh.glsl
Source Files
wave.c
wave.c Start Page
(Global Scope)
main(int argc, char ** argv)
223 glutPostRedisplay();
224 }
225
226 int main(int argc, char** argv)
227 {
228 int i,j;
229
230 /* flat mesh */
231
232 for(i=0;i<N;i++) for(j=0;j<N;j++) data[i][j]=0.0;
233
234 glutInit(&argc, argv);
235 glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
236 glutInitWindowSize(512, 512);
237 glutCreateWindow("WAVE GLSL example");
238 glutDisplayFunc(draw);
239 glutReshapeFunc(reshape);
240 glutKeyboardFunc(keyboard);
241 glutIdleFunc(idle);
242
243 init();
244 glewInit();
245 initShader("vmesh.glsl", "fPassthrough.glsl");
246
247 glutMainLoop();
248 }
249
Server Explorer
Toolbox
Solution Ex... Class View Property M...
Output
Show output from: Debug
Code Definition Window Call Browser Output Pending Checkins
Ready
Ln 243 Col 12 Ch 12 INS




```
// wave.c

/* sets up flat mesh */
/* sets up elapsed time parameter for use by shaders */
/* vertex shader varies height of mesh sinusoidally */
/* uses a pass through fragment shader */


#include <stdio.h>
#include <stdlib.h>


#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glew.h>
#include <GL/glut.h>
#endif

#define N 64
```

```
const GLdouble nearVal     = 1.0; /* near distance */
const GLdouble farVal      = 20.0; /* far distance */
GLuint         program     = 0; /* program object id */
GLint          timeParam;

GLchar *ebuffer; /* buffer for error messages */
GLsizei elength; /* length of error message */

GLfloat data[N][N]; /* array of data heights */

/* shader reader */
/* creates null terminated string from file */

static char* readShaderSource(const char* shaderFile)
{
    FILE* fp = fopen(shaderFile, "rb");

    char* buf;
    long size;

    if(fp==NULL) return NULL;
    fseek(fp, 0L, SEEK_END);
    size = ftell(fp);
    fseek(fp, 0L, SEEK_SET);
    buf = (char*) malloc((size+1) * sizeof(char));
    fread(buf, 1, size, fp);
      buf[size] = '\0';
    fclose(fp);
    return buf;
}

/* standard OpenGL initialization */

static void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glColor3f(0.0, 0.0, 0.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-0.75,0.75,-0.75,0.75,-5.5,5.5);
}

/* GLSL initialization */

static void initShader(const GLchar* vShaderFile, const GLchar*
fShaderFile)
{
    GLint status;
    GLchar *vSource, *fSource;
    GLuint vShader, fShader;

    /* read shader files */

    vSource = readShaderSource(vShaderFile);
    if(vSource==NULL)
    {
```

```
    printf( "Failed to read vertex shader\n");
    exit(EXIT_FAILURE);
}

fSource = readShaderSource(fShaderFile);
if(fSource==NULL)
{
    printf("Failed to read fragment shader");
   exit(EXIT_FAILURE);
}

/* create program and shader objects */

vShader = glCreateShader(GL_VERTEX_SHADER);
fShader = glCreateShader(GL_FRAGMENT_SHADER);
program = glCreateProgram();

/* attach shaders to the program object */

glAttachShader(program, vShader);
glAttachShader(program, fShader);

/* read shaders */

glShaderSource(vShader, 1, (const GLchar**) &vSource, NULL);
glShaderSource(fShader, 1, (const GLchar**) &fSource, NULL);

/* compile vertex shader shader */

glCompileShader(vShader);

/* error check */

glGetShaderiv(vShader, GL_COMPILE_STATUS, &status);
if(status==GL_FALSE)
{
   printf("Failed to compile the vertex shader.\n");
   glGetShaderiv(vShader, GL_INFO_LOG_LENGTH, &elength);
   ebuffer = malloc(elength*sizeof(char));
   glGetShaderInfoLog(vShader, elength, NULL, ebuffer);
   printf("%s\n", ebuffer);
  exit(EXIT_FAILURE);
}

/* compile fragment  shader shader */

glCompileShader(fShader);

/* error check */

glGetShaderiv(fShader, GL_COMPILE_STATUS, &status);
if(status==GL_FALSE)
{
   printf("Failed to compile the fragment shader.\n");
   glGetShaderiv(fShader, GL_INFO_LOG_LENGTH, &elength);
   ebuffer = malloc(elength*sizeof(char));
   glGetShaderInfoLog(fShader, elength, NULL, ebuffer);
```

```
        printf("%s\n", ebuffer);
        exit(EXIT_FAILURE);
    }

    /* link  and error check */

    glLinkProgram(program);
    glGetProgramiv(program, GL_LINK_STATUS, &status);
    if(status==GL_FALSE)
    {
        printf("Failed to link program object.\n");
        glGetProgramiv(program, GL_INFO_LOG_LENGTH, &elength);
        ebuffer = malloc(elength*sizeof(char));
        glGetProgramInfoLog(program, elength, &elength, ebuffer);
        printf("%s\n", ebuffer);
        exit(EXIT_FAILURE);
    }

    /* use program object */

    glUseProgram(program);

    /* set up uniform parameter */

    timeParam = glGetUniformLocation(program, "time");
}

void mesh()
{
    int i,j;
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(2.0, 2.0, 2.0, 0.5, 0.0, 0.5, 0.0, 1.0, 0.0);
    for(i=0; i<N; i++) for(j=0; j<N;j++)
    {
        glColor3f(1.0, 1.0, 1.0);
        glBegin(GL_POLYGON);
            glVertex3f((float)i/N, data[i][j], (float)j/N);
            glVertex3f((float)i/N, data[i][j], (float)(j+1)/N);
            glVertex3f((float)(i+1)/N, data[i][j], (float)(j+1)/N);
            glVertex3f((float)(i+1)/N, data[i][j], (float)(j)/N);
        glEnd();
        glColor3f(0.0, 0.0, 0.0);
        glBegin(GL_LINE_LOOP);
            glVertex3f((float)i/N, data[i][j], (float)j/N);
            glVertex3f((float)i/N, data[i][j], (float)(j+1)/N);
            glVertex3f((float)(i+1)/N, data[i][j], (float)(j+1)/N);
            glVertex3f((float)(i+1)/N, data[i][j], (float)(j)/N);
        glEnd();
    }
}

static void draw(void)
{
    /* send elapsed time to shaders */

    glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));
```

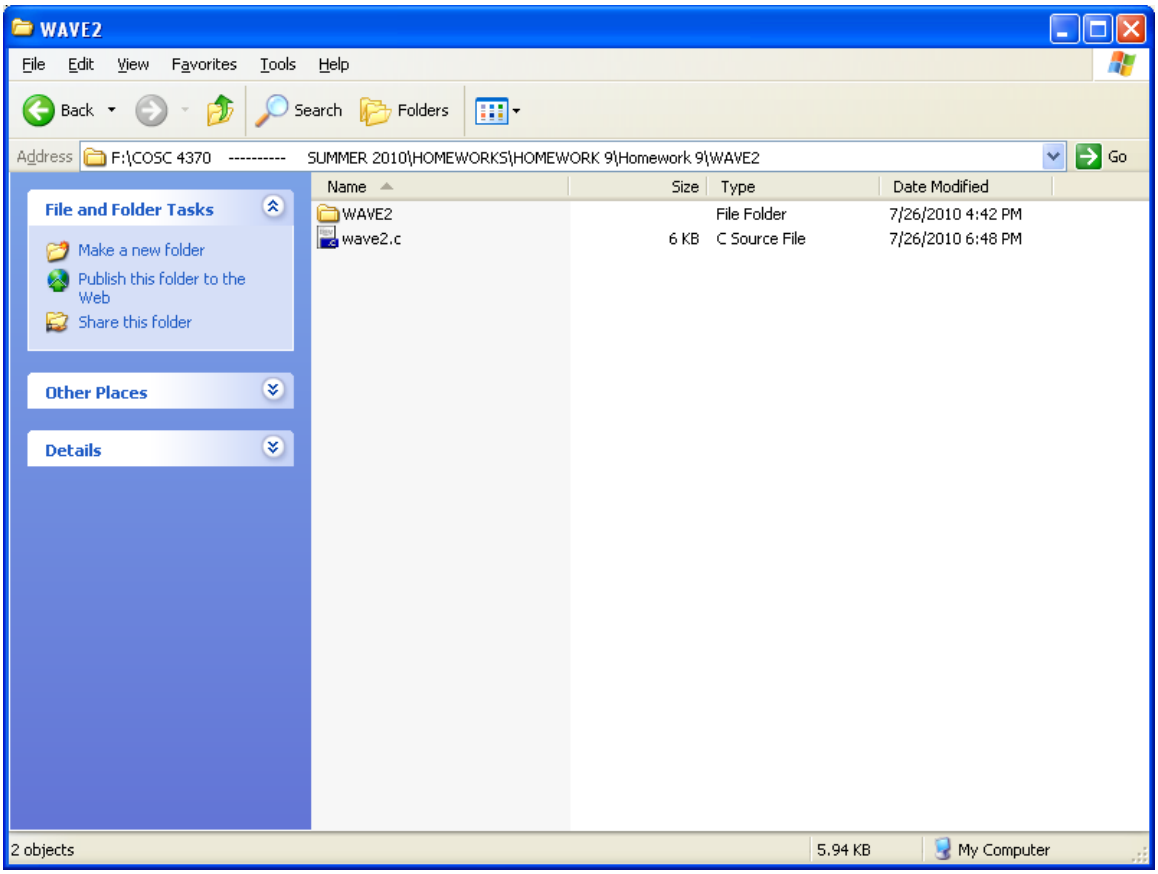
```
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    mesh();
    glutSwapBuffers();
}

static void reshape(int w, int h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-0.75,0.75,-0.75,0.75,-5.5,5.5);

    glViewport(0, 0, w, h);
    glutPostRedisplay();
}

static void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 27:
    case 'Q':
    case 'q':
        exit(EXIT_SUCCESS);
        break;
    default:
        break;
    }
}

void idle()
{
   glUniform1f(timeParam, (GLfloat) glutGet(GLUT_ELAPSED_TIME));
   glutPostRedisplay();
}

int main(int argc, char** argv)
{
    int i,j;

    /* flat mesh */

    for(i=0;i<N;i++) for(j=0;j<N;j++) data[i][j]=0.0;

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
    glutInitWindowSize(512, 512);
    glutCreateWindow("WAVE GLSL example");
    glutDisplayFunc(draw);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutIdleFunc(idle);

    init();
      glewInit();
    initShader("vmesh.glsl", "fPassthrough.glsl");

    glutMainLoop();
}
```

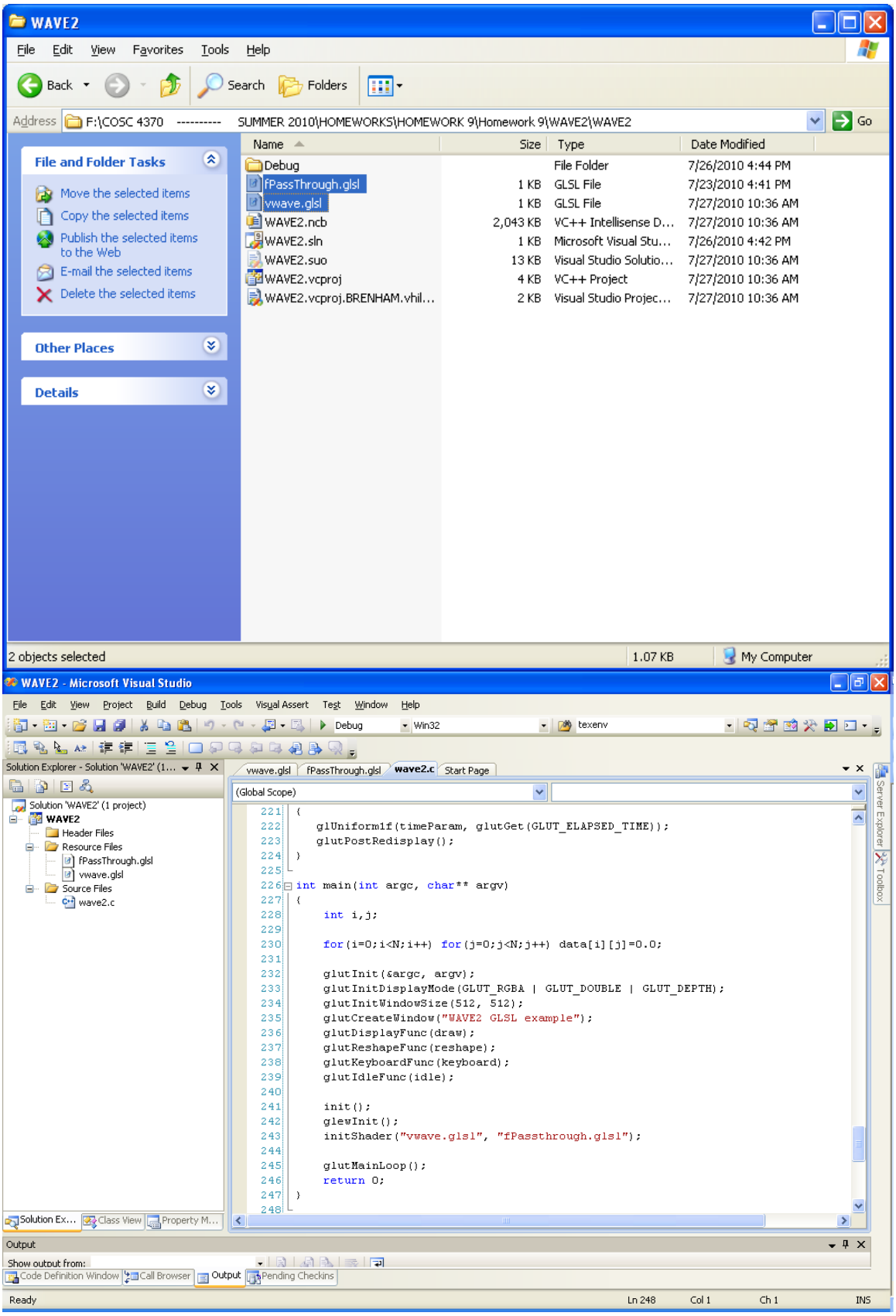
// vmesh.glsl

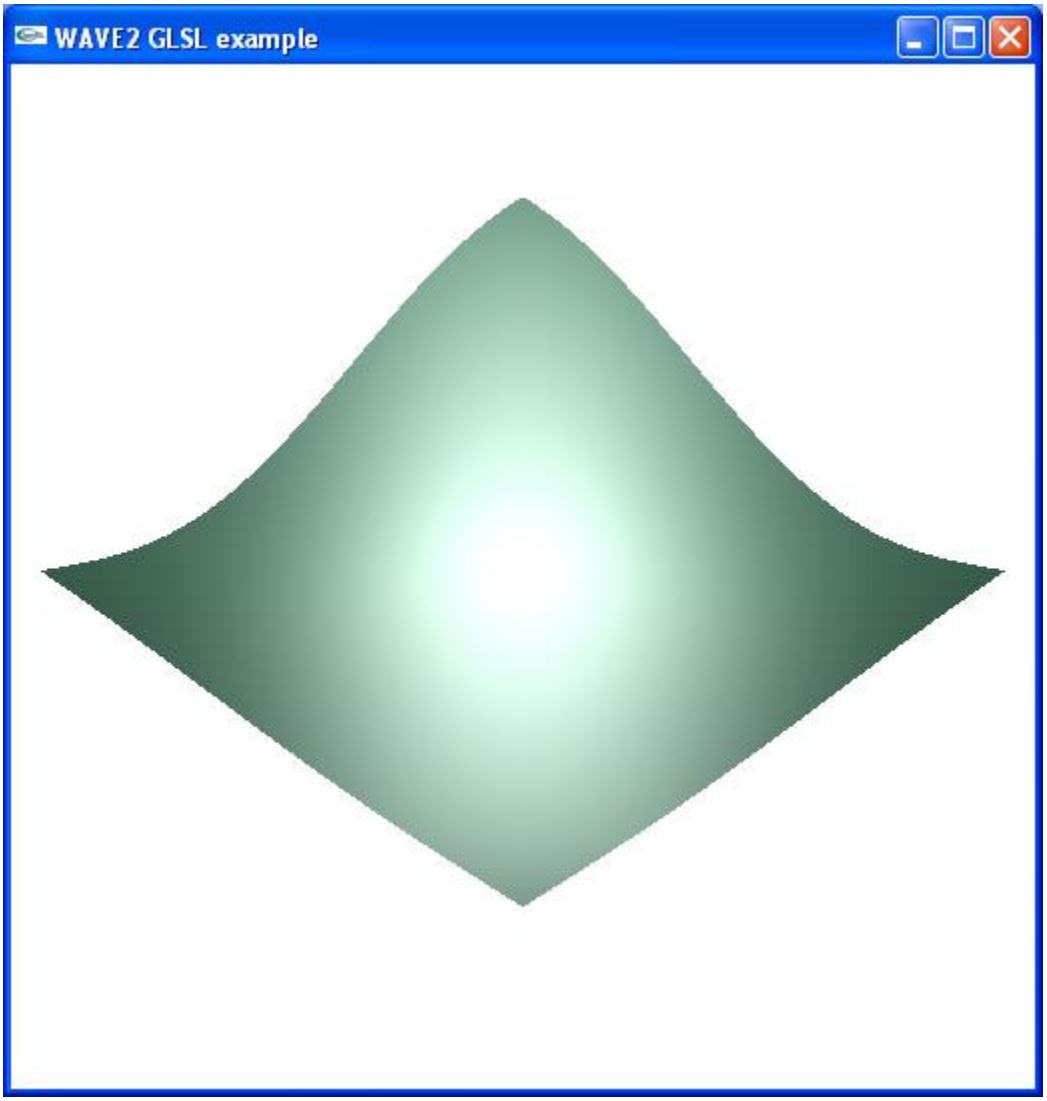
```
uniform float time; /* in milliseconds */

void main()
{
    float s;
    vec4 t = gl_Vertex;
    t.y =
0.1*sin(0.001*time+5.0*gl_Vertex.x)*sin(0.001*time+5.0*gl_Vertex.z);
    gl_Position = gl_ModelViewProjectionMatrix * t;
    gl_FrontColor = gl_Color;
}
```

```
// fPassThrough.glsl
// Pass through fragment shader.

void main()
{
     gl_FragColor = gl_Color;
}
```

8. WAVE2


WAVE2
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\WAVE2
Go
File and Folder Tasks
Make a new folder
Publish this folder to the Web
Share this folder
Other Places
Details
Name Size Type Date Modified
WAVE2 File Folder 7/26/2010 4:42 PM
wave2.c 6 KB C Source File 7/26/2010 6:48 PM
2 objects
5.94 KB
My Computer


WAVE2
File Edit View Favorites Tools Help
Back Search Folders
Address F:\COSC 4370 ---------- SUMMER 2010\HOMEWORKS\HOMEWORK 9\Homework 9\WAVE2\WAVE2
Go
File and Folder Tasks
Move the selected items
Copy the selected items
Publish the selected items to the Web
E-mail the selected items
Delete the selected items
Other Places
Details
Name Size Type Date Modified
Debug File Folder 7/26/2010 4:44 PM
fPassThrough.glsl 1 KB GLSL File 7/23/2010 4:41 PM
vwave.glsl 1 KB GLSL File 7/27/2010 10:36 AM
WAVE2.ncb 2,043 KB VC++ Intellisense D... 7/27/2010 10:36 AM
WAVE2.sln 1 KB Microsoft Visual Stu... 7/26/2010 4:42 PM
WAVE2.suo 13 KB Visual Studio Solutio... 7/27/2010 10:36 AM
WAVE2.vcproj 4 KB VC++ Project 7/27/2010 10:36 AM
WAVE2.vcproj.BRENHAM.vhil... 2 KB Visual Studio Projec... 7/27/2010 10:36 AM
2 objects selected
1.07 KB
My Computer
WAVE2 - Microsoft Visual Studio
File Edit View Project Build Debug Tools Visual Assert Test Window Help
Debug Win32 texenv
Solution Explorer - Solution 'WAVE2' (1...
Solution 'WAVE2' (1 project)
WAVE2
Header Files
Resource Files
fPassThrough.glsl
vwave.glsl
Source Files
wave2.c
vwave.glsl fPassThrough.glsl wave2.c Start Page
(Global Scope)
221 {
222 glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));
223 glutPostRedisplay();
224 }
225
226 int main(int argc, char** argv)
227 {
228 int i,j;
229
230 for(i=0;i<N;i++) for(j=0;j<N;j++) data[i][j]=0.0;
231
232 glutInit(&argc, argv);
233 glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
234 glutInitWindowSize(512, 512);
235 glutCreateWindow("WAVE2 GLSL example");
236 glutDisplayFunc(draw);
237 glutReshapeFunc(reshape);
238 glutKeyboardFunc(keyboard);
239 glutIdleFunc(idle);
240
241 init();
242 glewInit();
243 initShader("vwave.glsl", "fPassthrough.glsl");
244
245 glutMainLoop();
246 return 0;
247 }
248
Solution Ex... Class View Property M...
Server Explorer
Toolbox
Output
Show output from:
Code Definition Window Call Browser Output Pending Checkins
Ready
Ln 248 Col 1 Ch 1 INS




```
// wave2.c

/* sets up flat mesh */
/* sets up elapsed time parameter for use by shaders */
/* uses a modified phong vertex shader and a pass through fragment
shader */
/* vertex shader varies height of mesh sinusoidally */

#include <stdio.h>
#include <stdlib.h>
#include <GL/glew.h>
#include <GL/glut.h>

#define N 64

const GLdouble nearVal     = 1.0; /* near distance */
const GLdouble farVal      = 20.0; /* far distance */
const GLfloat  lightPos[4] = {0.0f, 3.0f, -3.0f, 1.0f}; /* light
position */
GLuint         program     = 0; /* program object id */
GLint          timeParam;
```

```
GLchar *ebuffer; /* buffer for error messages */
GLsizei elength; /* length of error message */

GLfloat data[N][N]; /* array of data heights */

/* shader reader */
/* creates null terminated string from file */

static char* readShaderSource(const char* shaderFile)
{
   FILE* fp = fopen(shaderFile, "rb");

    char* buf;
    long size;

    if(fp==NULL) return NULL;
    fseek(fp, 0L, SEEK_END);
    size = ftell(fp);
    fseek(fp, 0L, SEEK_SET);
    buf = (char*) malloc((size+1) * sizeof(char));
    fread(buf, 1, size, fp);
      buf[size] = '\0';
    fclose(fp);
    return buf;
}

/* standard OpenGL initialization */

static void init()
{
    const float meshColor[]     = {0.3f, 0.5f, 0.4f, 1.0f};
    const float meshSpecular[]  = {0.8f, 0.8f, 0.8f, 1.0f};
    const float meshShininess[] = {80.0f};

    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, meshColor);
    glMaterialfv(GL_FRONT, GL_SPECULAR, meshSpecular);
    glMaterialfv(GL_FRONT, GL_SHININESS, meshShininess);

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-0.75,0.75,-0.75,0.75,-5.5,5.5);

    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
}

/* GLSL initialization */

static void initShader(const GLchar* vShaderFile, const GLchar*
fShaderFile)
{
    GLint status;
```

```
GLchar *vSource, *fSource;
GLuint vShader, fShader;

/* read shader files */

vSource = readShaderSource(vShaderFile);
if(vSource==NULL)
{
    printf( "Failed to read vertex shaderi\n");
    exit(EXIT_FAILURE);
}

fSource = readShaderSource(fShaderFile);
if(fSource==NULL)
{
    printf("Failed to read fragment shader");
    exit(EXIT_FAILURE);
}

/* create program and shader objects */

vShader = glCreateShader(GL_VERTEX_SHADER);
fShader = glCreateShader(GL_FRAGMENT_SHADER);
program = glCreateProgram();

/* attach shaders to the program object */

glAttachShader(program, vShader);
glAttachShader(program, fShader);

/* read shaders */

glShaderSource(vShader, 1, (const GLchar**) &vSource, NULL);
glShaderSource(fShader, 1, (const GLchar**) &fSource, NULL);

/* compile vertex shader shader */

glCompileShader(vShader);

/* error check */

glGetShaderiv(vShader, GL_COMPILE_STATUS, &status);
if(status==GL_FALSE)
{
   printf("Failed to compile the vertex shader.\n");
   glGetShaderiv(vShader, GL_INFO_LOG_LENGTH, &elength);
   ebuffer = malloc(elength*sizeof(char));
   glGetShaderInfoLog(vShader, elength, NULL, ebuffer);
   printf("%s\n", ebuffer);
   exit(EXIT_FAILURE);
}

/* compile fragment  shader shader */

glCompileShader(fShader);

/* error check */
```

```
    glGetShaderiv(fShader, GL_COMPILE_STATUS, &status);
    if(status==GL_FALSE)
    {
       printf("Failed to compile the fragment shader.\n");
       glGetShaderiv(fShader, GL_INFO_LOG_LENGTH, &elength);
       ebuffer = malloc(elength*sizeof(char));
       glGetShaderInfoLog(fShader, elength, NULL, ebuffer);
       printf("%s\n", ebuffer);
       exit(EXIT_FAILURE);
    }

    /* link  and error check */

    glLinkProgram(program);
    glGetProgramiv(program, GL_LINK_STATUS, &status);
    if(status==GL_FALSE)
    {
       printf("Failed to link program object.\n");
       glGetProgramiv(program, GL_INFO_LOG_LENGTH, &elength);
       ebuffer = malloc(elength*sizeof(char));
       glGetProgramInfoLog(program, elength, &elength, ebuffer);
       printf("%s\n", ebuffer);
       exit(EXIT_FAILURE);
    }

    /* use program object */

    glUseProgram(program);

    /* set up uniform parameter */

    timeParam = glGetUniformLocation(program, "time");
}

void mesh()
{
    int i,j;
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(2.0, 2.0, 2.0, 0.5, 0.0, 0.5, 0.0, 1.0, 0.0);
    glNormal3f(0.0,1.0,0.0);
    glBegin(GL_QUADS);
    for(i=0; i<N; i++) for(j=0; j<N;j++)
    {
       glVertex3f((float)i/N, data[i][j], (float)j/N);
       glVertex3f((float)i/N, data[i][j], (float)(j+1)/N);
       glVertex3f((float)(i+1)/N, data[i][j], (float)(j+1)/N);
       glVertex3f((float)(i+1)/N, data[i][j], (float)(j)/N);
    }
    glEnd();
}

static void draw(void)
{
    /* send elapsed time to shaders */
```

```
    glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    mesh();
    glutSwapBuffers();
}

static void reshape(int w, int h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-0.75,0.75,-0.75,0.75,-5.5,5.5);

    glViewport(0, 0, w, h);
    glutPostRedisplay();
}

static void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 27:
    case 'Q':
    case 'q':
        exit(EXIT_SUCCESS);
        break;
    default:
        break;
    }
}

void idle()
{
   glUniform1f(timeParam, glutGet(GLUT_ELAPSED_TIME));
   glutPostRedisplay();
}

int main(int argc, char** argv)
{
    int i,j;

    for(i=0;i<N;i++) for(j=0;j<N;j++) data[i][j]=0.0;

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(512, 512);
    glutCreateWindow("WAVE2 GLSL example");
    glutDisplayFunc(draw);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutIdleFunc(idle);

    init();
    glewInit();
    initShader("vwave.glsl", "fPassthrough.glsl");

    glutMainLoop();
    return 0;
```

```
}

// vwave.glsl

uniform float time;
void main()
{
    float s;
    vec4 t = gl_Vertex;
    t.y =
0.1*sin(0.001*time+5.0*gl_Vertex.x)*sin(0.001*time+5.0*gl_Vertex.z);
    // t.y = 0.1*sin(0.01*time*gl_Vertex.x)*sin(0.01*time*gl_Vertex.z);
    gl_Position = gl_ModelViewProjectionMatrix * t;
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    vec4 eyePosition = gl_ModelViewMatrix * gl_Vertex;
    vec4 eyeLightPos = gl_LightSource[0].position;
    vec3 N = normalize(gl_NormalMatrix * gl_Normal);
    vec3 L = normalize(eyeLightPos.xyz - eyePosition.xyz);
    vec3 E = -normalize(eyePosition.xyz);
    vec3 H = normalize(L + E);
    float Kd = max(dot(L, N), 0.0);
    float Ks = pow(max(dot(N, H), 0.0), gl_FrontMaterial.shininess);
    float Ka = 0.0;

    ambient = Ka*gl_FrontLightProduct[0].ambient;
    diffuse = Kd*gl_FrontLightProduct[0].diffuse;
    specular = Ks*gl_FrontLightProduct[0].specular;
    gl_FrontColor = ambient+diffuse+specular;
}


// fPassThrough.glsl
// Pass through fragment shader.

void main()
{
     gl_FragColor = gl_Color;
}
```